DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy`

Feature	Description
	• `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the project was approved.

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_3:__ "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

your neighborhood, and your school are all helpful.

- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

  For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

In [1]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

```
C:\Users\samar\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; al
iasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

In [2]:

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [3]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
```

```
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[4]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of `project_subject_categories`

In [5]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

1.3 preprocessing of `project_subject_subcategories`

In [6]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
```

```
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

1.3b preprocessing of `project_grade_category`

In [7]:

```
project_data['project_grade_category'].values
```

Out[7]:

```
array(['Grades PreK-2', 'Grades 6-8', 'Grades 6-8', ..., 'Grades PreK-2',
       'Grades 3-5', 'Grades 6-8'], dtype=object)
```

In [8]:

```
prj_grade_cat = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

prj_grade_cat_list = []
for i in prj_grade_cat:
    for j in i.split(' '): # it will split by space
        j=j.replace('Grades','') # if we have the words "Grades" we are going to replace it with ''
(i.e removing 'Grades')
    prj_grade_cat_list.append(j.strip())

project_data['clean_grade'] = prj_grade_cat_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_grade'].values:
    my_counter.update(word.split())

prj_grade_cat_dict = dict(my_counter)
sorted_prj_grade_cat_dict = dict(sorted(prj_grade_cat_dict.items(), key=lambda kv: kv[1]))
```

In [9]:

```
project_data['clean_grade'].values
```

Out[9]:

```
array(['PreK-2', '6-8', '6-8', ..., 'PreK-2', '3-5', '6-8'], dtype=object)
```

1.3c preprocessing of `teacher_prefix`

In [10]:

```
project_data['teacher_prefix'].values
```

Out[10]:

```
array(['Mrs.', 'Mr.', 'Ms.', ..., 'Mrs.', 'Mrs.', 'Ms.'], dtype=object)
```

In [11]:

```
#tea_pfx_cat = list(project_data['teacher_prefix'].values)
tea_pfx_cat = list(project_data['teacher_prefix'].astype(str).values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

##https://stackoverflow.com/questions/52736900/how-to-solve-the-attribute-error-float-object-has-n
o-attribute-split-in-pyth
#vectorizer.fit(project_data['teacher_prefix'].astype(str).values)

tea_pfx_cat_list = []
for i in tea_pfx_cat:
    #for j in i.split(' '): # it will split by space
    #j=j.replace('.','') # if we have the words "Grades" we are going to replace it with ''(i.e re
moving 'Grades')
    i=i.replace('.','') # if we have the words "Grades" we are going to replace it with ''(i.e remo
ving 'Grades')
    i=i.replace('nan','') # if we have the words "Grades" we are going to replace it with ''(i.e re
moving 'Grades')
    tea_pfx_cat_list.append(i.strip())

project_data['clean_tea_pfx'] = tea_pfx_cat_list
project_data.drop(['teacher_prefix'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_tea_pfx'].values:
    my_counter.update(word.split())

tea_pfx_cat_dict = dict(my_counter)
sorted_tea_pfx_cat_dict = dict(sorted(tea_pfx_cat_dict.items(), key=lambda kv: kv[1]))
```

In [12]:

```
project_data['clean_tea_pfx'].values
```

Out[12]:

```
array(['Mrs', 'Mr', 'Ms', ..., 'Mrs', 'Mrs', 'Ms'], dtype=object)
```

1.3 Text preprocessing

In [13]:

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [14]:

```
project_data.head(2)
```

Out[14]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_title	projec
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	My stu Englisl that ar
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	Our st arrive school lea...

1.4.2.3 Using Pretrained Models: TFIDF weighted W2V

In [15]:

```
## printing some random reviews
#print(project_data['essay'].values[0])
#print("="*50)
#print(project_data['essay'].values[150])
#print("="*50)
#print(project_data['essay'].values[1000])
#print("="*50)
#print(project_data['essay'].values[20000])
#print("="*50)
#print(project_data['essay'].values[99999])
#print("="*50)
```

In [16]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [17]:

```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out
for my students. I teach in a Title I school where most of the students receive free or reduced pr
ice lunch.  Despite their disabilities and limitations, my students love coming to school and come
eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to gr
oove and move as you were in a meeting? This is how my kids feel all the time. The want to be able
to move as they learn or so they say.Wobble chairs are the answer and I love then because they dev
elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l
earn through games, my kids do not want to sit and do worksheets. They want to learn to count by j
umping and playing. Physical engagement is the key to our success. The number toss and color and s
hape mats can make that happen. My students will forget they are doing work and just have the fun
a 6 year old deserves.nannan
==================================================
```

In [18]:

```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir hardest working past their limitations.     The materials we have are the ones I seek out for
my students. I teach in a Title I school where most of the students receive free or reduced price
lunch.  Despite their disabilities and limitations, my students love coming to school and come eag
er to learn and explore.Have you ever felt like you had ants in your pants and you needed to groov
e and move as you were in a meeting? This is how my kids feel all the time. The want to be able to
move as they learn or so they say.Wobble chairs are the answer and I love then because they develo
p their core, which enhances gross motor and in Turn fine motor skills.   They also want to learn t
hrough games, my kids do not want to sit and do worksheets. They want to learn to count by jumping
and playing. Physical engagement is the key to our success. The number toss and color and shape ma
ts can make that happen. My students will forget they are doing work and just have the fun a 6 yea
r old deserves.nannan
```

In [19]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays cognitiv
e delays gross fine motor delays to autism They are eager beavers and always strive to work their
hardest working past their limitations The materials we have are the ones I seek out for my studen
ts I teach in a Title I school where most of the students receive free or reduced price lunch
Despite their disabilities and limitations my students love coming to school and come eager to lea
rn and explore Have you ever felt like you had ants in your pants and you needed to groove and mov
e as you were in a meeting This is how my kids feel all the time The want to be able to move as th
ey learn or so they say Wobble chairs are the answer and I love then because they develop their co
re which enhances gross motor and in Turn fine motor skills They also want to learn through games
my kids do not want to sit and do worksheets They want to learn to count by jumping and playing Ph
ysical engagement is the key to our success The number toss and color and shape mats can make that
happen My students will forget they are doing work and just have the fun a 6 year old deserves nan
nan
```

In [20]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
```

```
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [21]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248
[00:57<00:00, 1909.83it/s]
```

In [22]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[22]:

```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gros
s fine motor delays autism they eager beavers always strive work hardest working past limitations
the materials ones i seek students i teach title i school students receive free reduced price lunc
h despite disabilities limitations students love coming school come eager learn explore have ever
felt like ants pants needed groove move meeting this kids feel time the want able move learn say w
obble chairs answer i love develop core enhances gross motor turn fine motor skills they also want
learn games kids not want sit worksheets they want learn count jumping playing physical engagement
key success the number toss color shape mats make happen my students forget work fun 6 year old de
serves nannan'
```

In [23]:

```
# Suggestion 5.you can try improving the score using feature engineering hacks.Try including lengt
h,summary
# and observe the results and re-submit the assignment.

# https://stackoverflow.com/questions/18827198/python-count-number-of-words-in-a-list-strings
preprocessed_essays_wc = []
for item in preprocessed_essays:
    preprocessed_essays_wc.append(len(item.split()))

print(preprocessed_essays_wc[101])
```

```
141
```

In [24]:

```
# Suggestion 5.you can try improving the score using feature engineering hacks.Try including lengt
h,summary
```

```
# and observe the results and re-submit the assignment.

# https://stackoverflow.com/questions/18827198/python-count-number-of-words-in-a-list-strings
preprocessed_essays_len = []
for item in tqdm(preprocessed_essays):
    preprocessed_essays_len.append(len(item))

print(preprocessed_essays_len[101])
```

```
100%|████████████████████████████████████████████████████████████| 109248/109248
[00:00<00:00, 1952886.05it/s]
```

```
1041
```

In [25]:

```
print(preprocessed_essays_len[100])
```

```
1258
```

1.4 Preprocessing of `project_title`

In [26]:

```
# similarly you can preprocess the titles also
```

In [27]:

```
project_data.head(2)
```

Out[27]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_title	projec
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	My stu Englis that ar
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	Our st arrive school lea...

In [28]:

```
# printing some random essays.
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[1000])
print("="*50)
print(project_data['project_title'].values[20000])
print("="*50)
print(project_data['project_title'].values[99999])
print("="*50)
```

```
Educational Support for English Learners at Home
```

```
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
We Need To Move It While We Input It!
==================================================
Inspiring Minds by Enhancing the Educational Experience
==================================================
```

In [29]:

```
sent_title = decontracted(project_data['project_title'].values[20000])
print(sent_title)
print("="*50)
```

```
We Need To Move It While We Input It!
==================================================
```

In [30]:

```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent_title = sent_title.replace('\\r', ' ')
sent_title = sent_title.replace('\\"', ' ')
sent_title = sent_title.replace('\\n', ' ')
print(sent_title)
```

```
We Need To Move It While We Input It!
```

In [31]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent_title = re.sub('[^A-Za-z0-9]+', ' ', sent_title)
print(sent_title)
```

```
We Need To Move It While We Input It
```

In [32]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent_title = decontracted(sentance)
    sent_title = sent_title.replace('\\r', ' ')
    sent_title = sent_title.replace('\\"', ' ')
    sent_title = sent_title.replace('\\n', ' ')
    sent_title = re.sub('[^A-Za-z0-9]+', ' ', sent_title)
    # https://gist.github.com/sebleier/554280
    sent_title = ' '.join(e for e in sent_title.split() if e not in stopwords)
    preprocessed_title.append(sent_title.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████| 109248/109248
[00:02<00:00, 41782.96it/s]
```

In [33]:

```
# after preprocesing
preprocessed_title[10]
```

Out[33]:

```
'reading changes lives'
```

In [34]:

```
#print(project_data['project_title'])
print(preprocessed_title[101])
```

```
print(preprocessed_title[101])
```

```
fun physically fit
```

In [35]:

```
# Suggestion 5.you can try improving the score using feature engineering hacks.Try including lengt
h,summary
# and observe the results and re-submit the assignment.

# https://stackoverflow.com/questions/18827198/python-count-number-of-words-in-a-list-strings
preprocessed_title_wc = []
for item in preprocessed_title:
    preprocessed_title_wc.append(len(item.split()))

print(preprocessed_title_wc[101])
```

```
3
```

In [36]:

```
print(preprocessed_title[101])
print(len(preprocessed_title[101]))
```

```
fun physically fit
18
```

In [37]:

```
# Suggestion 5.you can try improving the score using feature engineering hacks.Try including lengt
h,summary
# and observe the results and re-submit the assignment.

# https://stackoverflow.com/questions/18827198/python-count-number-of-words-in-a-list-strings
preprocessed_title_len = []
for item in tqdm(preprocessed_title):
    #print(preprocessed_title)
    preprocessed_title_len.append(len(item))
    #print(len(preprocessed_title))

print(preprocessed_title_len[101])
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248
[00:00<00:00, 2330695.13it/s]
```

```
18
```

In [38]:

```
print(preprocessed_title_len[100])
```

```
45
```

In [39]:

```
print(project_data['project_resource_summary'][20])
```

```
My students need carpet in our library to brighten up our reading space.
```

In [40]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_prj_sum = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_resource_summary'].values):
    sent_title = decontracted(sentance)
    sent_title = sent_title.replace('\\r', ' ')
```

```
    sent_title = sent_title.replace('\\"', ' ')
    sent_title = sent_title.replace('\\n', ' ')
    sent_title = re.sub('[^A-Za-z0-9]+', ' ', sent_title)
    # https://gist.github.com/sebleier/554280
    sent_title = ' '.join(e for e in sent_title.split() if e not in stopwords)
    preprocessed_prj_sum.append(sent_title.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████| 109248/109248
[00:05<00:00, 18685.27it/s]
```

In [41]:

```
preprocessed_prj_sum[100]
```

Out[41]:

```
'my students need laptops printing abilities i would like students ability work projects print wor
ks research writings'
```

In [42]:

```
# Suggestion 5.you can try improving the score using feature engineering hacks.Try including lengt
h,summary
# and observe the results and re-submit the assignment.

# https://stackoverflow.com/questions/18827198/python-count-number-of-words-in-a-list-strings
preprocessed_prj_sum_wc = []
for item in preprocessed_prj_sum:
    preprocessed_prj_sum_wc.append(len(item.split()))

print(preprocessed_prj_sum_wc[100])
```

```
17
```

In [43]:

```
# Suggestion 5.you can try improving the score using feature engineering hacks.Try including lengt
h,summary
# and observe the results and re-submit the assignment.

# https://stackoverflow.com/questions/18827198/python-count-number-of-words-in-a-list-strings
preprocessed_prj_sum_len = []
for item in tqdm(preprocessed_prj_sum):
    preprocessed_prj_sum_len.append(len(item))

print(preprocessed_prj_sum_len[100])
```

```
100%|██████████████████████████████████████████████████████████████| 109248/109248
[00:00<00:00, 2375251.79it/s]
```

```
117
```

1.5 Preparing data for models

In [44]:

```
project_data.columns
```

Out[44]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state',
       'project_submitted_datetime', 'project_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'clean_grade',
       'clean_tea_pfx', 'essay'],
      dtype='object')
```

we are going to consider

```
- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)

- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical
```

1.5.1 Vectorizing Categorical data

- https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/

1.5.3 Vectorizing Numerical features

In [45]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [46]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 20)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'school_state'
 'project_submitted_datetime' 'project_title' 'project_essay_1'
 'project_essay_2' 'project_essay_3' 'project_essay_4'
 'project_resource_summary' 'teacher_number_of_previously_posted_projects'
 'project_is_approved' 'clean_categories' 'clean_subcategories'
 'clean_grade' 'clean_tea_pfx' 'essay' 'price' 'quantity']
```

Adding word count and length column as per suggestion 5

In [47]:

```
project_data['essay_wc'] = preprocessed_essays_wc
project_data['essay_len'] = preprocessed_essays_len

project_data['title_wc'] = preprocessed_title_wc
project_data['title_len'] = preprocessed_title_len

project_data['prj_res_sum_wc'] = preprocessed_prj_sum_wc
project_data['prj_res_sum_len'] = preprocessed_prj_sum_len
```

In [48]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 26)
--------------------------------------------------
```

```
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'school_state'
 'project_submitted_datetime' 'project_title' 'project_essay_1'
 'project_essay_2' 'project_essay_3' 'project_essay_4'
 'project_resource_summary' 'teacher_number_of_previously_posted_projects'
 'project_is_approved' 'clean_categories' 'clean_subcategories'
 'clean_grade' 'clean_tea_pfx' 'essay' 'price' 'quantity' 'essay_wc'
 'essay_len' 'title_wc' 'title_len' 'prj_res_sum_wc' 'prj_res_sum_len']
```

Assignment 4: Naive Bayes

1. **Apply Multinomial NaiveBayes on these feature sets**
   - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
   - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
2. **The hyper paramter tuning(find best Alpha)**
   - Find the best hyper parameter which will give the maximum AUC value
   - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
   - Find the best hyper paramter using k-fold cross validation or simple cross validation data
   - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning
3. **Feature importance**
   - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB and print their corresponding feature names
4. **Representation of results**
   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.
   - Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
   - Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.
5. **Conclusion**
   - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link

In [49]:

```
#taking 50K datapoint
#project_data50K=project_data[:50000]
#project_data1K=project_data[:1000]
#print(project_data50K.shape)
#print(project_data1K.shape)
```

In [50]:

```
# makins Xi as 19 column matrix, where we create the modle and Yi as single colum matrix as a clas
s label.
#y = project_data50K['project_is_approved'].values
#project_data50K.drop(['project_is_approved'], axis=1, inplace=True)
#print(y.shape)
#project_data50K.head(1)

y = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'], axis=1, inplace=True)
print(y.shape)
project_data.head(1)
```

```
(109248,)
```

Out[50]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_title	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	My stude English l that are

1 rows × 25 columns

In [51]:

```
#X = project_data50K
X = project_data
print(X.shape)

#X1K = project_data1K
#print(X1K.shape)
```

```
(109248, 25)
```

2. Naive Bayes

2.1 Splitting data into Train and cross validation(or test): Stratified Sampling

In [52]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [53]:

```
# train test split | https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
# spliting Xq and Yq in Train(further into Train and CV) and Test matrix
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)

print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)
```

```
(49041, 25) (49041,)
(24155, 25) (24155,)
(36052, 25) (36052,)
====================================================================================================
```

In [54]:

```
#from sklearn.model_selection import train_test_split
```

```
#from sklearn.model_selection import train_test_split
#X1K_train, X1K_test, y1K_train, y1K_test = train_test_split(X1K, y1K, test_size=0.33,
stratify=y1K)
#X1K_train, X1K_cv, y1K_train, y1K_cv = train_test_split(X1K_train, y1K_train, test_size=0.33, str
atify=y1K#_train)
#print(X1K_train.shape, y1K_train.shape)
#print(X1K_cv.shape, y1K_cv.shape)
#print(X1K_test.shape, y1K_test.shape)
#
#print("="*100)
```

2.1.1 Make Data Model Ready: encoding school_state categorical data

In [55]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,1))
# Don't use bigrams or trigrams. Limit ngram_range to just unigrams,
# because naive bayes assumes that features are independent to each other
vectorizer.fit(X_train['school_state'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer.transform(X_test['school_state'].values)

print("school_state After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
aa=vectorizer.get_feature_names()
print("="*100)
```

```
school_state After vectorizations
(49041, 51) (49041,)
(24155, 51) (24155,)
(36052, 51) (36052,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'k
s', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm',
'nv', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv
', 'wy']
==================================================================================================
```

2.1.2 Make Data Model Ready: encoding clean_categories

In [56]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary =list(sorted_cat_dict.keys()),lowercase =False,binary=True
)
#vectorizer = CountVectorizer(min_df=10,ngram_range=(1,1))
vectorizer.fit(X_train['clean_categories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_clean_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_cv_clean_ohe = vectorizer.transform(X_cv['clean_categories'].values)
X_test_clean_ohe = vectorizer.transform(X_test['clean_categories'].values)

print("clean_categories After vectorizations")
print(X_train_clean_ohe.shape, y_train.shape)
print(X_cv_clean_ohe.shape, y_cv.shape)
print(X_test_clean_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
b=vectorizer.get_feature_names()
```

```
clean_categories After vectorizations
(49041, 9) (49041,)
(24155, 9) (24155,)
(36052, 9) (36052,)
```

```
(36052, 9) (36052,)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
==================================================================================================
```

2.1.3 Make Data Model Ready: encoding clean_subcategories

In [57]:

```
from sklearn.feature_extraction.text import CountVectorizer
#vectorizer = CountVectorizer(vocabulary =list(sorted_sub_cat_dict.keys()),lowercase
=False,binary=True,min_df=10,ngram_range=(1,1))
vectorizer = CountVectorizer(vocabulary =list(sorted_sub_cat_dict.keys()),lowercase =False,binary=
True)
vectorizer.fit(X_train['clean_subcategories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_cleanSub_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
X_cv_cleanSub_ohe = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_cleanSub_ohe = vectorizer.transform(X_test['clean_subcategories'].values)

print("clean_subcategories After vectorizations")
print(X_train_cleanSub_ohe.shape, y_train.shape)
print(X_cv_cleanSub_ohe.shape, y_cv.shape)
print(X_test_cleanSub_ohe.shape, y_test.shape)
#print(vectorizer.get_feature_names())
print("="*100)
c=vectorizer.get_feature_names()
```

```
clean_subcategories After vectorizations
(49041, 30) (49041,)
(24155, 30) (24155,)
(36052, 30) (36052,)
==================================================================================================
```

2.1.4 Make Data Model Ready: encoding project_grade_category

In [58]:

```
print(X_train['clean_grade'].values)
```

```
['3-5' 'PreK-2' 'PreK-2' ... '6-8' 'PreK-2' 'PreK-2']
```

In [59]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary =list(sorted_prj_grade_cat_dict.keys()),lowercase =False,b
inary=True)
vectorizer.fit(X_train['clean_grade'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer.transform(X_train['clean_grade'].values)
X_cv_grade_ohe = vectorizer.transform(X_cv['clean_grade'].values)
X_test_grade_ohe = vectorizer.transform(X_test['clean_grade'].values)

print("project_grade_category After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
d=vectorizer.get_feature_names()
```

```
project_grade_category After vectorizations
(49041, 4) (49041,)
(24155, 4) (24155,)
(36052, 4) (36052,)
['9-12', '6-8', '3-5', 'PreK-2']
```

```
==========================================================================================================
```

2.1.5 Make Data Model Ready: encoding teacher_prefix

In [60]:

```
from sklearn.feature_extraction.text import CountVectorizer
#vectorizer = CountVectorizer(min_df=10,ngram_range=(1,1), max_features=5000)
vectorizer = CountVectorizer(vocabulary =list(sorted_tea_pfx_cat_dict.keys()),lowercase =False,bin
ary=True)
vectorizer.fit(X_train['clean_tea_pfx'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer.transform(X_train['clean_tea_pfx'].values)
X_cv_teacher_ohe = vectorizer.transform(X_cv['clean_tea_pfx'].values)
X_test_teacher_ohe = vectorizer.transform(X_test['clean_tea_pfx'].values)

print("teacher_prefix After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
e=vectorizer.get_feature_names()
```

```
teacher_prefix After vectorizations
(49041, 5) (49041,)
(24155, 5) (24155,)
(36052, 5) (36052,)
['Dr', 'Teacher', 'Mr', 'Ms', 'Mrs']
==========================================================================================================
```

2.1.6 Make Data Model Ready: encoding project_resource_summary

In [61]:

```
#vectorizer = CountVectorizer()
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,1))
vectorizer.fit(X_train['project_resource_summary'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_prjResSum_ohe = vectorizer.transform(X_train['project_resource_summary'].values)
X_cv_prjResSum_ohe = vectorizer.transform(X_cv['project_resource_summary'].values)
X_test_prjResSum_ohe = vectorizer.transform(X_test['project_resource_summary'].values)

print("project_resource_summary After vectorizations")
print(X_train_prjResSum_ohe.shape, y_train.shape)
print(X_cv_prjResSum_ohe.shape, y_cv.shape)
print(X_test_prjResSum_ohe.shape, y_test.shape)
#print(vectorizer.get_feature_names())
print("="*100)
f=vectorizer.get_feature_names()
print(type(f))
```

```
project_resource_summary After vectorizations
(49041, 3988) (49041,)
(24155, 3988) (24155,)
(36052, 3988) (36052,)
==========================================================================================================

<class 'list'>
```

2.1.7 Make Data Model Ready: encoding essay

In [62]:

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,1))
vectorizer.fit(X_train['essay'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow = vectorizer.transform(X_train['essay'].values)
X_cv_essay_bow = vectorizer.transform(X_cv['essay'].values)
X_test_essay_bow = vectorizer.transform(X_test['essay'].values)

print("Essay After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
print("="*100)
g=vectorizer.get_feature_names()
```

```
Essay After vectorizations
(49041, 12506) (49041,)
(24155, 12506) (24155,)
(36052, 12506) (36052,)
====================================================================================================
```

2.1.8 Make Data Model Ready: encoding project title

In [63]:

```
#vectorizer = CountVectorizer()
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,1))
vectorizer.fit(X_train['project_title'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_title_bow = vectorizer.transform(X_train['project_title'].values)
X_cv_title_bow = vectorizer.transform(X_cv['project_title'].values)
X_test_title_bow = vectorizer.transform(X_test['project_title'].values)

print("project_title After vectorizations")
print(X_train_title_bow.shape, y_train.shape)
print(X_cv_title_bow.shape, y_cv.shape)
print(X_test_title_bow.shape, y_test.shape)
#print(vectorizer.get_feature_names())
print("="*100)
k=vectorizer.get_feature_names()
```

```
project_title After vectorizations
(49041, 2092) (49041,)
(24155, 2092) (24155,)
(36052, 2092) (36052,)
====================================================================================================
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [64]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

2.2.1 Make Data Model Ready: encoding numerical | price

In [65]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['price'].values.reshape(-1,1))

X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(-1,1))
X_cv_price_norm = normalizer.transform(X_cv['price'].values.reshape(-1,1))
X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(-1,1))

print("Price After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
#h='price'
```

```
Price After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

2.2.2 Make Data Model Ready: encoding numerical| teacher_number_of_previously_posted_projects

In [66]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

X_train_TprevPrj_norm =
normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_cv_TprevPrj_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].val
ues.reshape(-1,1))
X_test_TprevPrj_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projects']
.values.reshape(-1,1))

print("teacher_number_of_previously_posted_projects After vectorizations")
print(X_train_TprevPrj_norm.shape, y_train.shape)
print(X_cv_TprevPrj_norm.shape, y_cv.shape)
print(X_test_TprevPrj_norm.shape, y_test.shape)
print("="*100)
```

```
teacher_number_of_previously_posted_projects After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

2.2.3 Make Data Model Ready: encoding numerical | quantity

In [67]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
```

```
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['quantity'].values.reshape(-1,1))

X_train_quantity_norm = normalizer.transform(X_train['quantity'].values.reshape(-1,1))
X_cv_quantity_norm = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))
X_test_quantity_norm = normalizer.transform(X_test['quantity'].values.reshape(-1,1))

print("quantity After vectorizations")
print(X_train_quantity_norm.shape, y_train.shape)
print(X_cv_quantity_norm.shape, y_cv.shape)
print(X_test_quantity_norm.shape, y_test.shape)
print("="*100)
#j='quantity'
```

```
quantity After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [68]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

Suggestion #5

we can try improving the score using feature engineering hacks.Try including length,summary and observe the results and re-submit the assignment.

2.3.3.1 Make Data Model Ready: encoding numerical | essay word

In [69]:

```
#print(X_train['quantity'].values)
#print(type(X_train['quantity'].values))
#print(X_train['quantity'].values.reshape(-1,1))
#
#print(type(preprocessed_essays_wc))
##print(preprocessed_essays_wc)
#
```

In [70]:

```
## https://stackoverflow.com/questions/5951135/how-to-save-a-list-as-numpy-array-in-python
#from numpy  import array
#pre_essays_wc_narr = array( preprocessed_essays_wc )
#print(type(pre_essays_wc_narr))
#print(pre_essays_wc_narr)
#print(pre_essays_wc_narr.reshape(-1,1))
```

In [71]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['essay_wc'].values.reshape(-1,1))

X_train_essay_wc_norm = normalizer.transform(X_train['essay_wc'].values.reshape(-1,1))
X_cv_essay_wc_norm = normalizer.transform(X_cv['essay_wc'].values.reshape(-1,1))
X_test_essay_wc_norm = normalizer.transform(X_test['essay_wc'].values.reshape(-1,1))

print("essay_wc After vectorizations")
print(X_train_essay_wc_norm.shape, y_train.shape)
print(X_cv_essay_wc_norm.shape, y_cv.shape)
print(X_test_essay_wc_norm.shape, y_test.shape)
print("="*100)
```

```
essay_wc After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

2.3.3.2 Make Data Model Ready: encoding numerical | essay length

In [72]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['essay_len'].values.reshape(-1,1))

X_train_essay_len_norm = normalizer.transform(X_train['essay_len'].values.reshape(-1,1))
X_cv_essay_len_norm = normalizer.transform(X_cv['essay_len'].values.reshape(-1,1))
X_test_essay_len_norm = normalizer.transform(X_test['essay_len'].values.reshape(-1,1))

print("essay_len After vectorizations")
print(X_train_essay_len_norm.shape, y_train.shape)
print(X_cv_essay_len_norm.shape, y_cv.shape)
print(X_test_essay_len_norm.shape, y_test.shape)
print("="*100)
```

```
essay_len After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

2.3.3.3 Make Data Model Ready: encoding numerical | title wordcount

In [73]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
```

```
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['title_wc'].values.reshape(-1,1))

X_train_title_wc_norm = normalizer.transform(X_train['title_wc'].values.reshape(-1,1))
X_cv_title_wc_norm = normalizer.transform(X_cv['title_wc'].values.reshape(-1,1))
X_test_title_wc_norm = normalizer.transform(X_test['title_wc'].values.reshape(-1,1))

print("title_wc After vectorizations")
print(X_train_title_wc_norm.shape, y_train.shape)
print(X_cv_title_wc_norm.shape, y_cv.shape)
print(X_test_title_wc_norm.shape, y_test.shape)
print("="*100)
```

```
title_wc After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

2.3.3.4 Make Data Model Ready: encoding numerical | title length

In [74]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['title_len'].values.reshape(-1,1))

X_train_title_len_norm = normalizer.transform(X_train['title_len'].values.reshape(-1,1))
X_cv_title_len_norm = normalizer.transform(X_cv['title_len'].values.reshape(-1,1))
X_test_title_len_norm = normalizer.transform(X_test['title_len'].values.reshape(-1,1))

print("title_len After vectorizations")
print(X_train_title_len_norm.shape, y_train.shape)
print(X_cv_title_len_norm.shape, y_cv.shape)
print(X_test_title_len_norm.shape, y_test.shape)
print("="*100)
```

```
title_len After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

2.3.3.5 Make Data Model Ready: encoding numerical | project resource summary wordcount

In [75]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['prj_res_sum_wc'].values.reshape(-1,1))

X_train_prj_res_sum_wc_norm = normalizer.transform(X_train['prj_res_sum_wc'].values.reshape(-1,1))
X_cv_prj_res_sum_wc_norm = normalizer.transform(X_cv['prj_res_sum_wc'].values.reshape(-1,1))
X_test_prj_res_sum_wc_norm = normalizer.transform(X_test['prj_res_sum_wc'].values.reshape(-1,1))

print("prj_res_sum_wc After vectorizations")
```

```
print("prj_res_sum_wc After vectorizations")
print(X_train_prj_res_sum_wc_norm.shape, y_train.shape)
print(X_cv_prj_res_sum_wc_norm.shape, y_cv.shape)
print(X_test_prj_res_sum_wc_norm.shape, y_test.shape)
print("="*100)
```

```
prj_res_sum_wc After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

2.3.3.6 Make Data Model Ready: encoding numerical | project resource summary length

In [76]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['prj_res_sum_len'].values.reshape(-1,1))

X_train_prj_res_sum_len_norm = normalizer.transform(X_train['prj_res_sum_len'].values.reshape(-1,1)
)
X_cv_prj_res_sum_len_norm = normalizer.transform(X_cv['prj_res_sum_len'].values.reshape(-1,1))
X_test_prj_res_sum_len_norm = normalizer.transform(X_test['prj_res_sum_len'].values.reshape(-1,1))

print("prj_res_sum_len After vectorizations")
print(X_train_prj_res_sum_len_norm.shape, y_train.shape)
print(X_cv_prj_res_sum_len_norm.shape, y_cv.shape)
print(X_test_prj_res_sum_len_norm.shape, y_test.shape)
print("="*100)
```

```
prj_res_sum_len After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

Adding all the features

In [77]:

```
h=['price','quantity','teacher_number_of_previously_posted_projects']
print(type(h))
```

```
<class 'list'>
```

In [78]:

```
l=['essay_wc','essay_len','title_wc','title_len','prj_res_sum_wc','prj_res_sum_len']
print(type(l))
```

```
<class 'list'>
```

In [79]:

```
feature_list=[]
print(feature_list)
feature_list=aa+b+c+d+e+f+g+k+h+l
#print(feature_list)
print(len(feature_list))
```

```
[]
18694
```

2.4 Appling NB() on different kind of featurization as mentioned in the instructions

Apply Naive Bayes on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instrucations

1. **Apply Multinomial NaiveBayes on these feature sets**
    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

2.4.1 Applying Naive Bayes on BOW, SET 1

- Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)

In [80]:

```
# Please write all the code with proper documentation
```

In [81]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_bow = hstack((X_train_essay_bow, X_train_title_bow, X_train_state_ohe, X_train_clean_ohe,
X_train_cleanSub_ohe, X_train_grade_ohe, X_train_teacher_ohe, X_train_prjResSum_ohe,
X_train_quantity_norm, X_train_TprevPrj_norm, X_train_price_norm)).tocsr()
X_cv_bow = hstack((X_cv_essay_bow, X_cv_title_bow, X_cv_state_ohe, X_cv_clean_ohe,
X_cv_cleanSub_ohe, X_cv_grade_ohe, X_cv_teacher_ohe, X_cv_prjResSum_ohe, X_cv_quantity_norm, X_cv_T
prevPrj_norm, X_cv_price_norm)).tocsr()
X_te_bow = hstack((X_test_essay_bow, X_test_title_bow , X_test_state_ohe, X_test_clean_ohe, X_test_
cleanSub_ohe, X_test_grade_ohe, X_test_teacher_ohe, X_test_prjResSum_ohe, X_test_quantity_norm,
X_test_TprevPrj_norm, X_test_price_norm)).tocsr()

print("Final Data matrix | BOW")
print(X_tr_bow.shape, y_train.shape)
print(X_cv_bow.shape, y_cv.shape)
print(X_te_bow.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix | BOW
(49041, 18688) (49041,)
(24155, 18688) (24155,)
(36052, 18688) (36052,)
====================================================================================================
```

After sugestion #5

In [82]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_sug5_bow = hstack((X_train_essay_bow, X_train_title_bow, X_train_state_ohe, X_train_clean_ohe
, X_train_cleanSub_ohe, X_train_grade_ohe, X_train_teacher_ohe, X_train_prjResSum_ohe,
X_train_quantity_norm, X_train_TprevPrj_norm,
X_train_price_norm,X_train_essay_wc_norm,X_train_essay_len_norm,X_train_title_wc_norm,X_train_title
_len_norm,X_train_prj_res_sum_wc_norm,X_train_prj_res_sum_len_norm)).tocsr()
X_cv_sug5_bow = hstack((X_cv_essay_bow, X_cv_title_bow, X_cv_state_ohe, X_cv_clean_ohe, X_cv_cleanS
ub_ohe, X_cv_grade_ohe, X_cv_teacher_ohe, X_cv_prjResSum_ohe, X_cv_quantity_norm, X_cv_TprevPrj_nor
m,
X_cv_price_norm,X_cv_essay_wc_norm,X_cv_essay_len_norm,X_cv_title_wc_norm,X_cv_title_len_norm,X_cv_
prj_res_sum_wc_norm,X_cv_prj_res_sum_len_norm)).tocsr()
```

```
X_te_sug5_bow = hstack((X_test_essay_bow, X_test_title_bow , X_test_state_ohe, X_test_clean_ohe, X_
test_cleanSub_ohe, X_test_grade_ohe, X_test_teacher_ohe, X_test_prjResSum_ohe,
X_test_quantity_norm, X_test_TprevPrj_norm,
X_test_price_norm,X_test_essay_wc_norm,X_test_essay_len_norm,X_test_title_wc_norm,X_test_title_len_
norm,X_test_prj_res_sum_wc_norm,X_test_prj_res_sum_len_norm)).tocsr()

print("Final Data matrix | BOW")
print(X_tr_sug5_bow.shape, y_train.shape)
print(X_cv_sug5_bow.shape, y_cv.shape)
print(X_te_sug5_bow.shape, y_test.shape)
print("="*100)
```

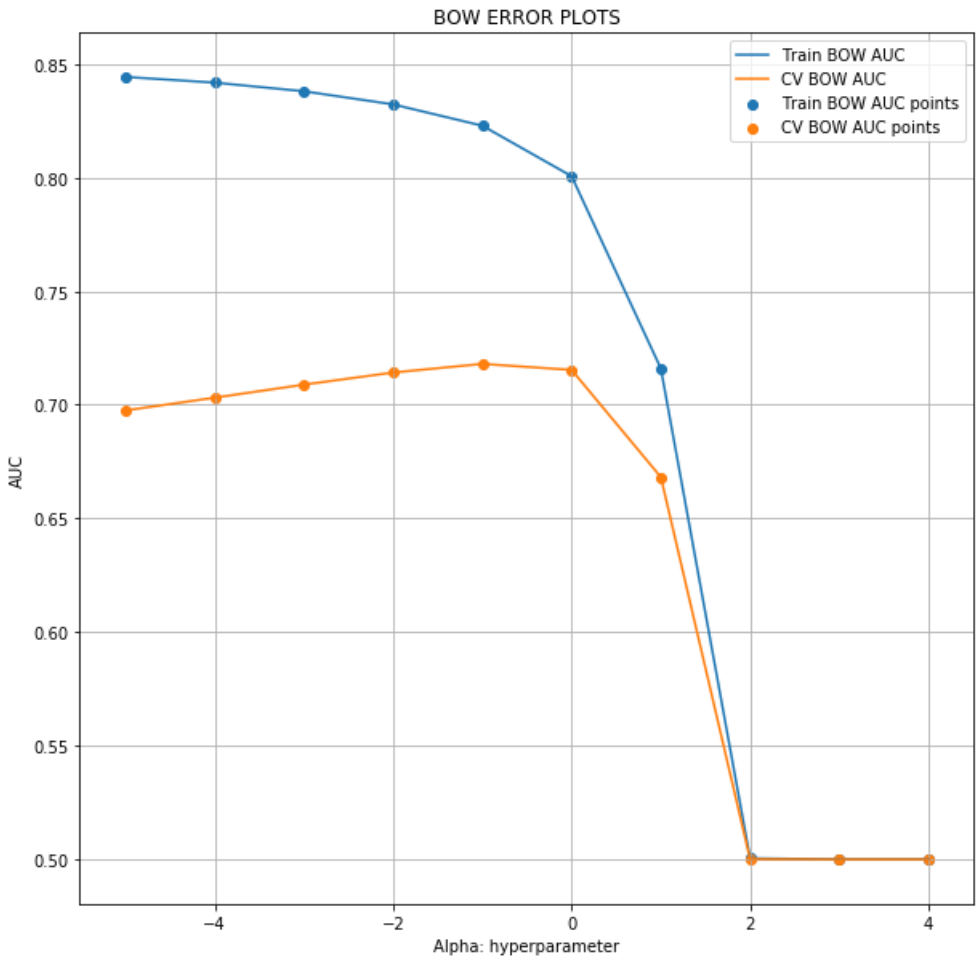
```
Final Data matrix | BOW
(49041, 18694) (49041,)
(24155, 18694) (24155,)
(36052, 18694) (36052,)
====================================================================================================
```

In [83]:

```
#import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or no
n-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_bow_auc = []
cv_bow_auc = []

a = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0, 10, 100, 1000, 10000]
for i in a:
    clf = MultinomialNB(alpha = i,class_prior=[0.5, 0.5])
    clf.fit(X_tr_bow, y_train)

    #y_train_bow_pred = batch_predict(clf, X_tr_bow)
    #y_cv_bow_pred = batch_predict(clf, X_cv_bow)

    y_train_bow_pred = clf.predict_proba(X_tr_bow)[:,1]
    y_cv_bow_pred = clf.predict_proba(X_cv_bow)[:,1]

    #y_train_bow_pred = clf.predict_log_proba(X_tr_bow)[:,1]
    #y_cv_bow_pred = clf.predict_log_proba(X_cv_bow)[:,1]

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html
    train_bow_auc.append(roc_auc_score(y_train,y_train_bow_pred))
    cv_bow_auc.append(roc_auc_score(y_cv, y_cv_bow_pred))

###############################################################
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/

import pickle
with open('pickleFiles\\train_alpha_bow_auc', 'wb') as ff:
    pickle.dump(train_bow_auc, ff)
import pickle
with open('pickleFiles\\cv_alpha_bow_auc', 'wb') as ff:
    pickle.dump(cv_bow_auc, ff)
###############################################################
# change each element of list in its log value python
# https://stackoverflow.com/questions/47582264/python-how-to-convert-a-list-to-loglist

# Suggestion 2
```

```
# Use log(alpha) on your X-axis, so that it will be more readable and we can understand what actua
lly happening there.
#https://docs.scipy.org/doc/numpy/reference/generated/numpy.log10.html

from math import log
alpha = [np.log10(x) for x in a]
print(alpha)
#print(type(alpha))

#https://stackoverflow.com/questions/332289/how-do-you-change-the-size-of-figures-drawn-with-matpl
otlib

plt.figure(figsize=(10,10))
plt.plot(alpha, train_bow_auc, label='Train BOW AUC')
plt.plot(alpha, cv_bow_auc, label='CV BOW AUC')

plt.scatter(alpha, train_bow_auc, label='Train BOW AUC points')
plt.scatter(alpha, cv_bow_auc, label='CV BOW AUC points')
# plt.xscale('log')
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("BOW ERROR PLOTS")

plt.grid(True)
plt.show()
```

```
[-5.0, -4.0, -3.0, -2.0, -1.0, 0.0, 1.0, 2.0, 3.0, 4.0]
```



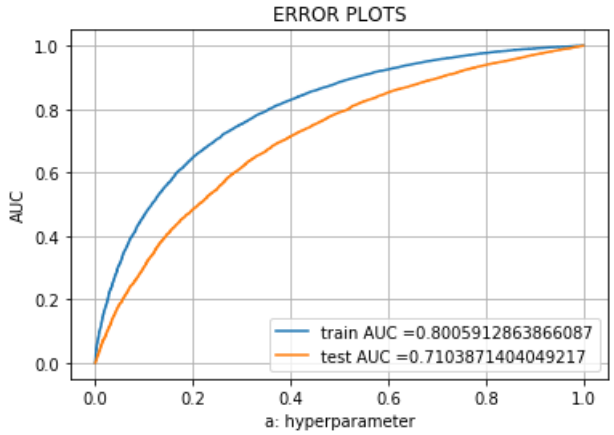
In [84]:

```
best_bow_a = 1
```

In [85]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
```

```
from sklearn.metrics import roc_curve, auc


neigh = MultinomialNB(alpha = best_bow_a)
neigh.fit(X_tr_bow, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_bow_pred = neigh.predict_proba(X_tr_bow)[:,1]
y_test_bow_pred = neigh.predict_proba(X_te_bow)[:,1]

train_bow_fpr, train_bow_tpr, tr_bow_thresholds = roc_curve(y_train, y_train_bow_pred)
test_bow_fpr, test_bow_tpr, te_bow_thresholds = roc_curve(y_test, y_test_bow_pred)

plt.plot(train_bow_fpr, train_bow_tpr, label="train AUC ="+str(auc(train_bow_fpr, train_bow_tpr)))
plt.plot(test_bow_fpr, test_bow_tpr, label="test AUC ="+str(auc(test_bow_fpr, test_bow_tpr)))
plt.legend()
plt.xlabel("a: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```



In [86]:

```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Vectorizer", "Algorithm", "Hyper parameter", "Train AUC", "Test AUC"]

x.add_row(["BOW", "Naives Bayes", 0.01, 0.831201, 0.70756 ])
x.add_row(["BOW", "Naives Bayes", 0.1, 0.82166, 0.712704 ])
x.add_row(["BOW", "Naives Bayes", 0, 0.8469, 0.67336 ])
x.add_row(["BOW (Best)", "Naives Bayes", 1, 0.800979, 0.71314 ])
x.add_row(["BOW", "Naives Bayes", 10, 0.711502, 0.671248 ])

print(x)
```

```
+------------+-------------+-----------------+-----------+----------+
| Vectorizer |  Algorithm  | Hyper parameter | Train AUC | Test AUC |
+------------+-------------+-----------------+-----------+----------+
|    BOW     | Naives Bayes |       0.01      |  0.831201 | 0.70756  |
|    BOW     | Naives Bayes |       0.1       |  0.82166  | 0.712704 |
|    BOW     | Naives Bayes |        0        |   0.8469  | 0.67336  |
| BOW (Best) | Naives Bayes |        1        |  0.800979 | 0.71314  |
|    BOW     | Naives Bayes |        10       |  0.711502 | 0.671248 |
+------------+-------------+-----------------+-----------+----------+
```

In [87]:

```
##print(confusion_matrix(y_train, predict(y_train_bow_pred, tr_bow_thresholds, train_bow_fpr, trai
n_bow_tpr)))
#
#print(y_train_bow_pred)  # [3.08371403e-03 1.31379587e-03 9.99999962e-01 ... 5.82935232e-04 7.875
```

```
92264e-02 9.99998321e-01]
#print(type(y_train_bow_pred)) # <class 'numpy.ndarray'>
#print(tr_bow_thresholds) # [2.00000000e+00 1.00000000e+00 1.00000000e+00 ... 1.80849883e-12 1.273
50934e-12 9.37174258e-18]
#print(type(tr_bow_thresholds)) # <class 'numpy.ndarray'>
#print(train_bow_fpr) # [0.00000000e+00 6.73309992e-04 6.73309992e-04 ... 9.98653380e-01 9.9865338
0e-01 1.00000000e+00]
#print(type(train_bow_fpr)) # <class 'numpy.ndarray'>
#print(train_#bow_tpr) # [0.         0.00730506 0.00780968 ... 0.99997597 1.         1.        ]
#print(type(train_bow_tpr)) # <class 'numpy.ndarray'>
```

In [88]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(fpr*(1-tpr))]
    #t = threshould[np.argmax(tpr*(1-fpr))]

    #print(proba)
    #print(threshould)
    #print(fpr)
    #print(tpr)
    #print("fpr*(1-tpr)",fpr*(1-tpr))
    #print("tpr*(1-fpr)",tpr*(1-fpr))
    #print(np.argmax(fpr*(1-tpr)))                  # 3712 when fpr*(1-tpr)
    #print(np.argmax(tpr*(1-fpr)))                  # 4033 when tpr*(1-fpr)
    #
    ##print(threshould[np.argmax(fpr*(1-tpr))])
    #print("t:",t)
    #
    ## t,threshould is 0.9176618541120772, when fpr*(1-tpr)
    ## t,threshould is 0.8477895257412753, when tpr*(1-fpr)
    ## (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [89]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_bow_pred, tr_bow_thresholds, train_bow_fpr,
train_bow_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_bow_pred, te_bow_thresholds, test_bow_fpr,
test_bow_tpr)))
```

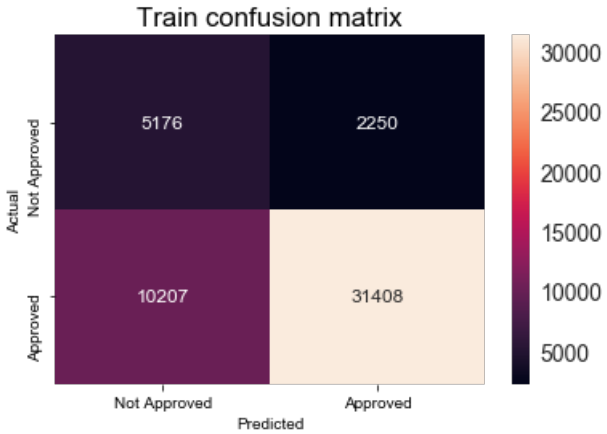
```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5320195426384736 for threshold 0.758
[[ 5176  2250]
 [10207 31408]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4369213882992951 for threshold 0.93
[[ 3544  1915]
 [10159 20434]]
```

In [90]:

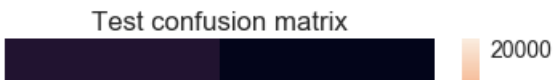
```
import seaborn as snTr
import seaborn as snTe
import pandas as pdH
import matplotlib.pyplot as pltTr
```

```
import matplotlib.pyplot as pltTe

# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
arrayTr=confusion_matrix(y_train, predict(y_train_bow_pred, tr_bow_thresholds, train_bow_fpr,
train_bow_tpr))
df_cmTr = pdH.DataFrame(arrayTr,range(2),range(2))
#print(arrayTr)
# https://stackoverflow.com/questions/32723798/how-do-i-add-a-title-to-seaborn-heatmap
axTr = pltTr.axes()

snTr.set(font_scale=1.4)#for label size

# https://seaborn.pydata.org/generated/seaborn.heatmap.html

snTr.heatmap(df_cmTr, annot=True,annot_kws={"size": 12},fmt="d",ax=axTr)# font size, format in
digit

labels=['Not Approved','Approved']
axTr.set_xticklabels(labels)
axTr.set_yticklabels(labels)
#Suggestion 4.Label confusion matrix heatmap with actual and predicted labels.
pltTr.title("Train confusion matrix")
pltTr.xlabel("Predicted")
pltTr.ylabel("Actual")
pltTr.show()

# https://stackoverflow.com/questions/50947776/plot-two-seaborn-heatmap-graphs-side-by-side
#fig, ax =plt.subplots(1,1)

# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
arrayTe=confusion_matrix(y_test, predict(y_test_bow_pred, te_bow_thresholds, test_bow_fpr,
test_bow_tpr))
df_cmTe = pdH.DataFrame(arrayTe,range(2),range(2))

axTe = pltTe.axes()

snTe.set(font_scale=1.4)#for label size

# https://seaborn.pydata.org/generated/seaborn.heatmap.html
snTe.heatmap(df_cmTe, annot=True,annot_kws={"size": 12},fmt="d",ax=axTe)# font size, format in
digit

#Suggestion 4.Label confusion matrix heatmap with actual and predicted labels.
axTe.set_xticklabels(labels)
axTe.set_yticklabels(labels)
pltTe.title("Test confusion matrix")
pltTe.xlabel("Predicted")
pltTe.ylabel("Actual")
pltTe.show()
```

```
the maximum value of tpr*(1-fpr) 0.5320195426384736 for threshold 0.758
```




```
the maximum value of tpr*(1-fpr) 0.4369213882992951 for threshold 0.93
```



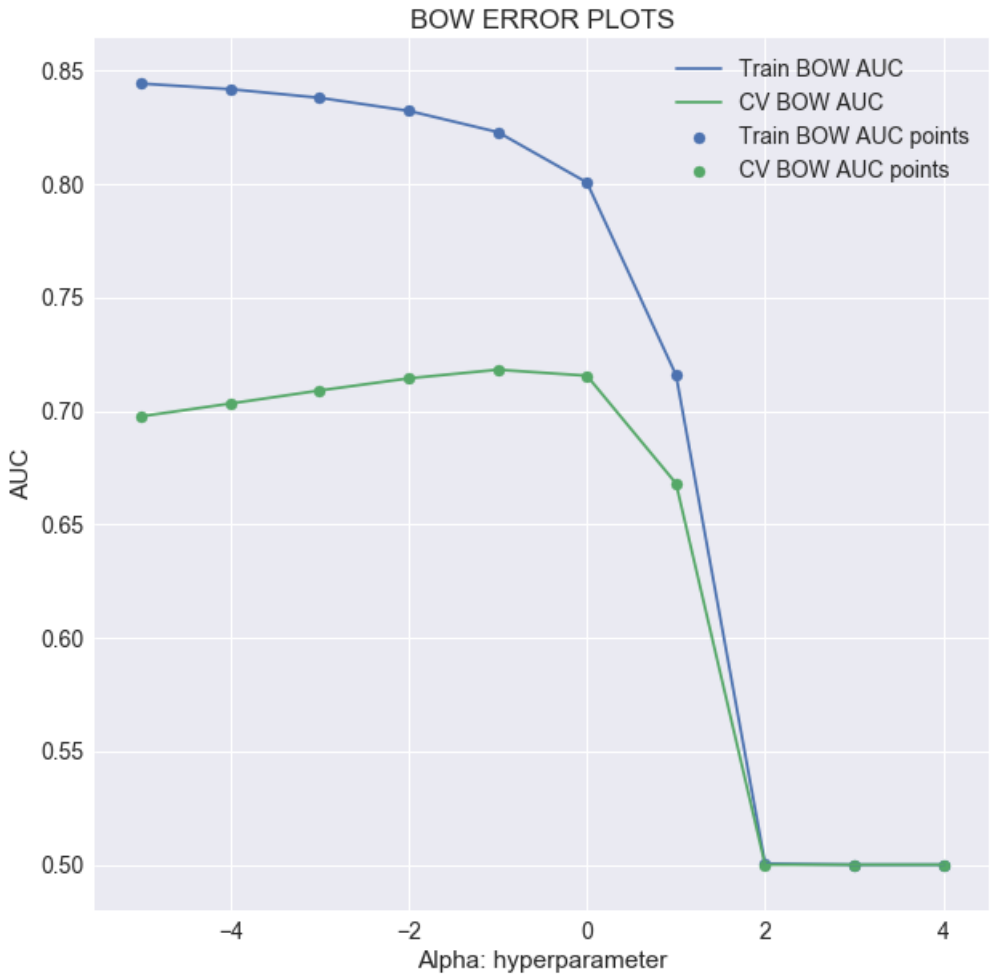
Suggestion #5

In [97]:

```
#import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or no
n-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_sug5_bow_auc = []
cv_sug5_bow_auc = []

a = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0, 10, 100, 1000, 10000]
for i in a:
    clf = MultinomialNB(alpha = i,class_prior = [0.5,0.5])
    clf.fit(X_tr_sug5_bow, y_train)

    y_train_sug5_bow_pred = clf.predict_proba(X_tr_sug5_bow)[:,1]
    y_cv_sug5_bow_pred = clf.predict_proba(X_cv_sug5_bow)[:,1]

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html
    train_sug5_bow_auc.append(roc_auc_score(y_train,y_train_sug5_bow_pred))
    cv_sug5_bow_auc.append(roc_auc_score(y_cv, y_cv_sug5_bow_pred))

# change each element of list in its log value python
# https://stackoverflow.com/questions/47582264/python-how-to-convert-a-list-to-loglist

# Suggestion 2
# Use log(alpha) on your X-axis, so that it will be more readable and we can understand what actua
lly happening there.
# #https://docs.scipy.org/doc/numpy/reference/generated/numpy.log10.html

from math import log
#alpha = [log(x) for x in a]
alpha = [np.log10(x) for x in a]
print(alpha)

#https://stackoverflow.com/questions/332289/how-do-you-change-the-size-of-figures-drawn-with-matpl
otlib

plt.figure(figsize=(10,10))

plt.plot(alpha, train_sug5_bow_auc, label='Train BOW AUC')
plt.plot(alpha, cv_sug5_bow_auc, label='CV BOW AUC')

plt.scatter(alpha, train_sug5_bow_auc, label='Train BOW AUC points')
```

```
plt.scatter(alpha, cv_sug5_bow_auc, label='CV BOW AUC points')
# plt.xscale('log')
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("BOW ERROR PLOTS")

plt.grid(True)
plt.show()
```

```
[-5.0, -4.0, -3.0, -2.0, -1.0, 0.0, 1.0, 2.0, 3.0, 4.0]
```



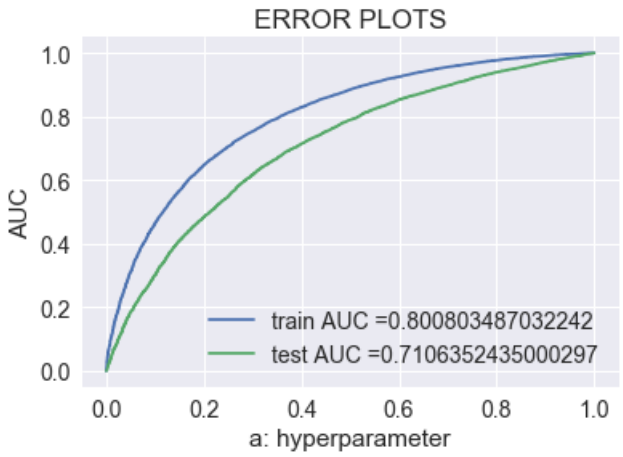
In [98]:

```
best_bow_a = 1
```

In [99]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = MultinomialNB(alpha = best_bow_a)
neigh.fit(X_tr_sug5_bow, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

#y_train_bow_pred = batch_predict(neigh, X_tr_bow)
#y_test_bow_pred = batch_predict(neigh, X_te_bow)
#print((X_tr_bow)[:,1])
y_train_sug5_bow_pred = neigh.predict_proba(X_tr_sug5_bow)[:,1]
y_test_sug5_bow_pred = neigh.predict_proba(X_te_sug5_bow)[:,1]

train_bow_fpr, train_bow_tpr, tr_bow_thresholds = roc_curve(y_train, y_train_sug5_bow_pred)
test bow fpr, test bow tpr, te bow thresholds = roc curve(y test, y test sug5 bow pred)
```

```
plt.plot(train_bow_fpr, train_bow_tpr, label="train AUC ="+str(auc(train_bow_fpr, train_bow_tpr)))
plt.plot(test_bow_fpr, test_bow_tpr, label="test AUC ="+str(auc(test_bow_fpr, test_bow_tpr)))
plt.legend()
plt.xlabel("a: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```



In [100]:

```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Vectorizer", "Algorithm", "Hyper parameter", "Train AUC", "Test AUC"]

x.add_row(["BOW (added feature)", "Naives Bayes", 0.01, 0.831337 , 0.707801 ])
x.add_row(["BOW (added feature)", "Naives Bayes", 0.1, 0.82182 , 0.71296 ])
x.add_row(["BOW (added feature)", "Naives Bayes", 0, 0.84706 , 0.67357 ])
x.add_row(["BOW (added feature)(Best)", "Naives Bayes", 1, 0.800803 , 0.710635 ])
x.add_row(["BOW (added feature)", "Naives Bayes", 10, 0.71144 , 0.671143 ])

print(x)
```

```
+--------------------------+--------------+-----------------+-----------+----------+
|         Vectorizer       |  Algorithm   | Hyper parameter | Train AUC | Test AUC |
+--------------------------+--------------+-----------------+-----------+----------+
|    BOW (added feature)   | Naives Bayes |       0.01      |  0.831337 | 0.707801 |
|    BOW (added feature)   | Naives Bayes |       0.1       |  0.82182  | 0.71296  |
|    BOW (added feature)   | Naives Bayes |        0        |  0.84706  | 0.67357  |
| BOW (added feature)(Best) | Naives Bayes |        1        |  0.800803 | 0.710635 |
|    BOW (added feature)   | Naives Bayes |        10       |  0.71144  | 0.671143 |
+--------------------------+--------------+-----------------+-----------+----------+
```

In [102]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_sug5_bow_pred, tr_bow_thresholds, train_bow_fpr,
train_bow_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_sug5_bow_pred, te_bow_thresholds, test_bow_fpr,
test_bow_tpr)))
```

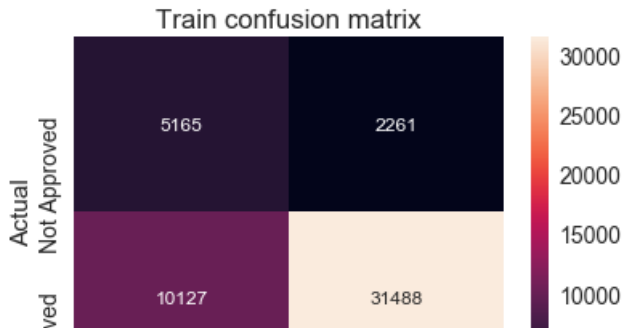
```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5321953296960302 for threshold 0.747
[[ 5165  2261]
 [10127 31488]]
```

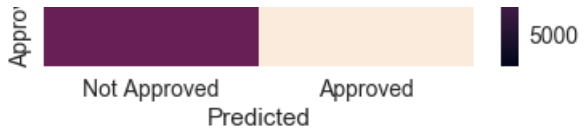
```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.43682435055923674 for threshold 0.933
[[ 3559  1900]
 [10230 20363]]
```

In [103]:

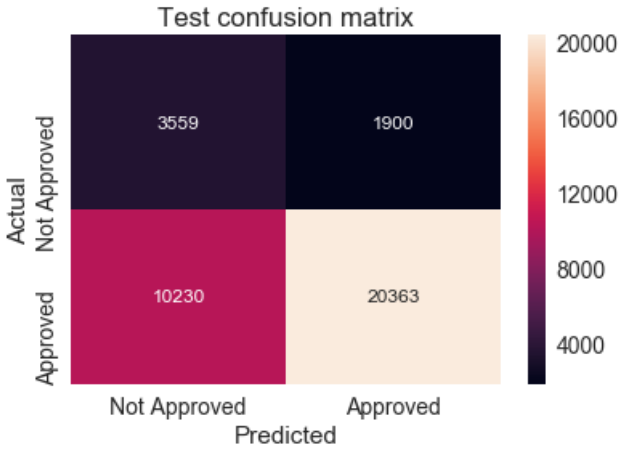
```
import seaborn as snTr
import seaborn as snTe
import pandas as pdH
import matplotlib.pyplot as pltTr
import matplotlib.pyplot as pltTe

# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
arrayTr=confusion_matrix(y_train, predict(y_train_sug5_bow_pred, tr_bow_thresholds, train_bow_fpr,
train_bow_tpr))
df_cmTr = pdH.DataFrame(arrayTr,range(2),range(2))
#print(arrayTr)
# https://stackoverflow.com/questions/32723798/how-do-i-add-a-title-to-seaborn-heatmap
axTr = pltTr.axes()

snTr.set(font_scale=1.4)#for label size

# https://seaborn.pydata.org/generated/seaborn.heatmap.html

snTr.heatmap(df_cmTr, annot=True,annot_kws={"size": 12},fmt="d",ax=axTr)# font size, format in
digit

labels=['Not Approved','Approved']
axTr.set_xticklabels(labels)
axTr.set_yticklabels(labels)
#Suggestion 4.Label confusion matrix heatmap with actual and predicted labels.
pltTr.title("Train confusion matrix")
pltTr.xlabel("Predicted")
pltTr.ylabel("Actual")
pltTr.show()

# https://stackoverflow.com/questions/50947776/plot-two-seaborn-heatmap-graphs-side-by-side
#fig, ax =plt.subplots(1,1)

# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
arrayTe=confusion_matrix(y_test, predict(y_test_sug5_bow_pred, te_bow_thresholds, test_bow_fpr,
test_bow_tpr))
df_cmTe = pdH.DataFrame(arrayTe,range(2),range(2))

axTe = pltTe.axes()

snTe.set(font_scale=1.4)#for label size

# https://seaborn.pydata.org/generated/seaborn.heatmap.html
snTe.heatmap(df_cmTe, annot=True,annot_kws={"size": 12},fmt="d",ax=axTe)# font size, format in
digit

#Suggestion 4.Label confusion matrix heatmap with actual and predicted labels.
axTe.set_xticklabels(labels)
axTe.set_yticklabels(labels)
pltTe.title("Test confusion matrix")
pltTe.xlabel("Predicted")
pltTe.ylabel("Actual")
pltTe.show()
```

```
the maximum value of tpr*(1-fpr) 0.5321953296960302 for threshold 0.747
```






```
the maximum value of tpr*(1-fpr) 0.43682435055923674 for threshold 0.933
```



2.4.1.1 Top 10 important features of positive class from SET 1

In [104]:

```
print(neigh.feature_log_prob_[1, :]) #probablity for positive features
```

```
[-11.63767104 -10.4261794   -9.00401913 ...  -5.66487015  -5.66487015
  -5.66487015]
```

In [105]:

```
print(len(feature_list))
print(feature_list[:10])
print(neigh.feature_log_prob_[0, :].shape)
print(neigh.feature_log_prob_[0, :]) #probablity for negative features
```

```
18694
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl']
(18694,)
[-11.54518934 -10.39778689  -9.21304545 ...  -5.62804433  -5.62804433
  -5.62804433]
```

In [106]:

```
pos_class_prob_sorted = neigh.feature_log_prob_[1, :].argsort()

# argsort() will return the indices of values from low probability to high probability.
# When you print feature names of these indices, these indices will return you the feature names w
ith low probability.
# So, please reverse the indices after argsort()
```

In [107]:

```
print(pos_class_prob_sorted)  # in low probablity to high
print(pos_class_prob_sorted[::-1]) # reverse ; now from high probablity to low

pos_class_prob_sorted=pos_class_prob_sorted[::-1]
print(pos_class_prob_sorted)
```

```
[14689 14688 14690 ... 11230   639 11384]
[11384   639 11230 ... 14690 14688 14689]
[11384   639 11230 ... 14690 14688 14689]
```

In [108]:

```
# https://imgur.com/a/1QObgQ2
# https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-bayes
print(np.take(feature_list, pos_class_prob_sorted[:10]))
```

```
['neighborhood' 'campus' 'nalbert' 'medicine' 'empower' 'boys' 'designer'
 'classes' 'nate' 'nalso']
```

Suggestion

In you data, you will be knowing hstacked features and their dimensions. Check the word in dimension and print that word. Suppose you have text+essay+categorial let text have dimension 120, essay 120, categorical 10 hstack in this way (text+essay+categorical) 120+120+10=250 dimensions Let important word indices be 10,196,243. for index 10 search in text , 196 in essay, 243 in categorical.

In [109]:

```
# https://cmdlinetips.com/2018/01/how-to-create-pandas-dataframe-from-multiple-lists/
Pos_Feature_dataFrame=pd.DataFrame({'Feature Word': feature_list,'Feature_Probablity' :
neigh.feature_log_prob_[1, :]})
```

In [110]:

```
Pos_Feature_dataFrame.head(10)
```

Out[110]:

	Feature Word	Feature_Probablity
0	ak	-11.637671
1	al	-10.426179
2	ar	-9.004019
3	az	-8.432091
4	ca	-11.983622
5	co	-13.210068
6	ct	-13.082234
7	dc	-13.410738
8	de	-13.662053
9	fl	-13.816203

In [111]:

```
# https://cmdlinetips.com/2018/02/how-to-sort-pandas-dataframe-by-columns-and-row/
Pos_Feature_dataFrame_Sorted=Pos_Feature_dataFrame.sort_values('Feature_Probablity',ascending=Fals
e)
Pos_Feature_dataFrame_Sorted.head(11)
```

Out[111]:

	Feature Word	Feature_Probablity
11384	neighborhood	-3.231371
639	campus	-3.391393
11230	nalbert	-3.479501
10830	medicine	-3.685733
7796	empower	-3.960883
5508	boys	-4.050451
7104	designer	-4.187345

	Feature Word	Feature_Probablity
774	classes	-4.315653
11268	nate	-4.362917
11235	nalso	-4.462347
12288	peg	-4.466021

2.4.1.2 Top 10 important features of negative class from SET 1

In [112]:

```
neg_class_prob_sorted = neigh.feature_log_prob_[0, :].argsort() # #probablity for negative
features, in low to high probablity
```

In [113]:

```
print(neg_class_prob_sorted)  # in low probablity to high
print(neg_class_prob_sorted[::-1]) # reverse ; now from high probablity to low

neg_class_prob_sorted=neg_class_prob_sorted[::-1]
print(neg_class_prob_sorted)
```

```
[11096 14448  6443 ... 11230   639 11384]
[11384   639 11230 ...  6443 14448 11096]
[11384   639 11230 ...  6443 14448 11096]
```

In [114]:

```
# https://imgur.com/a/1QObgQ2
# https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-bayes
print(np.take(feature_list, neg_class_prob_sorted[:10]))
```

```
['neighborhood' 'campus' 'nalbert' 'medicine' 'empower' 'boys' 'designer'
 'classes' 'nate' 'nalso']
```

Suggestion

In you data, you will be knowing hstacked features and their dimensions. Check the word in dimension and print that word. Suppose you have text+essay+categorial let text have dimension 120, essay 120, categorical 10 hstack in this way (text+essay+categorical) 120+120+10=250 dimensions Let important word indices be 10,196,243. for index 10 search in text , 196 in essay, 243 in categorical.

In [115]:

```
# https://cmdlinetips.com/2018/01/how-to-create-pandas-dataframe-from-multiple-lists/
Neg_Feature_dataFrame=pd.DataFrame({'Feature Word': feature_list,'Feature_Probablity' :
neigh.feature_log_prob_[0, :]})
```

In [116]:

```
Neg_Feature_dataFrame.head(10)
```

Out[116]:

	Feature Word	Feature_Probablity
0	ak	-11.545189
1	al	-10.397787
2	ar	-9.213045
3	az	-8.354713
4	ca	-11.449879
5	co	-12.931484

	Feature Word	Feature_Probablity
6	fe	-12.749162
7	dc	-13.442309
8	de	-13.442309
9	fl	-12.749162

In [117]:

```
# https://cmdlinetips.com/2018/02/how-to-sort-pandas-dataframe-by-columns-and-row/
Neg_Feature_dataFrame_Sorted=Neg_Feature_dataFrame.sort_values('Feature_Probablity',ascending=Fals
e)
Neg_Feature_dataFrame_Sorted.head(11)
```

Out[117]:

	Feature Word	Feature_Probablity
11384	neighborhood	-3.243891
639	campus	-3.395804
11230	nalbert	-3.511693
10830	medicine	-3.723546
7796	empower	-3.969887
5508	boys	-4.036841
7104	designer	-4.229140
774	classes	-4.307627
11268	nate	-4.344466
11235	nalso	-4.486001
12288	peg	-4.520718

2.4.2 Applying Naive Bayes on TFIDF, SET 2

In [118]:

```
# Please write all the code with proper documentation
```

1. Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

TFIDF Vectorizer on project_essay

In [119]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
Tfidf_vectorizer = TfidfVectorizer(min_df=10)

Tfidf_vectorizer.fit(X_train['essay'].values)

X_train_text_tfidf = Tfidf_vectorizer.transform(X_train['essay'].values)
X_cv_text_tfidf = Tfidf_vectorizer.transform(X_cv['essay'].values)
X_test_text_tfidf = Tfidf_vectorizer.transform(X_test['essay'].values)

##print("Shape of matrix after one hot encodig ",text_tfidf.shape)

print("Essay After vectorizations")
print(X_train_text_tfidf.shape, y_train.shape)
print(X_cv_text_tfidf.shape, y_cv.shape)
print(X_test_text_tfidf.shape, y_test.shape)
#print(Tfidf_vectorizer.get_feature_names())
print("="*100)
ii=Tfidf_vectorizer.get_feature_names()
```

```
Essay After vectorizations
(49041, 12506) (49041,)
(24155, 12506) (24155,)
(36052, 12506) (36052,)
==========================================================================================================
```

In [120]:

```
print(type(Tfidf_vectorizer.get_feature_names()))
#print(i)
print(type(ii))
```

```
<class 'list'>
<class 'list'>
```

TFIDF Vectorizer on project_title

In [121]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
Tfidf_vectorizer = TfidfVectorizer(min_df=10)

Tfidf_vectorizer.fit(X_train['project_title'].values)

X_train_title_tfidf = Tfidf_vectorizer.transform(X_train['project_title'].values)
X_cv_title_tfidf = Tfidf_vectorizer.transform(X_cv['project_title'].values)
X_test_title_tfidf = Tfidf_vectorizer.transform(X_test['project_title'].values)


##print("Shape of matrix after one hot encodig ",text_tfidf.shape)

print("project_title After vectorizations")
print(X_train_title_tfidf.shape, y_train.shape)
print(X_cv_title_tfidf.shape, y_cv.shape)
print(X_test_title_tfidf.shape, y_test.shape)
#print(Tfidf_vectorizer.get_feature_names())
print("="*100)
j=Tfidf_vectorizer.get_feature_names()
print(type(j))
```

```
project_title After vectorizations
(49041, 2092) (49041,)
(24155, 2092) (24155,)
(36052, 2092) (36052,)
==========================================================================================================

<class 'list'>
```

In [122]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_tfidf = hstack((X_train_text_tfidf, X_train_title_tfidf, X_train_state_ohe, X_train_clean_ohe
, X_train_cleanSub_ohe, X_train_grade_ohe, X_train_teacher_ohe, X_train_prjResSum_ohe,
X_train_quantity_norm, X_train_TprevPrj_norm, X_train_price_norm)).tocsr()
X_cv_tfidf = hstack((X_cv_text_tfidf, X_cv_title_tfidf, X_cv_state_ohe, X_cv_clean_ohe, X_cv_cleanS
ub_ohe, X_cv_grade_ohe, X_cv_teacher_ohe, X_cv_prjResSum_ohe, X_cv_quantity_norm, X_cv_TprevPrj_nor
m, X_cv_price_norm)).tocsr()
X_te_tfidf = hstack((X_test_text_tfidf, X_test_title_tfidf , X_test_state_ohe, X_test_clean_ohe, X_
test_cleanSub_ohe, X_test_grade_ohe, X_test_teacher_ohe, X_test_prjResSum_ohe,
X_test_quantity_norm, X_test_TprevPrj_norm, X_test_price_norm)).tocsr()

print("Final Data matrix | tfidf")
print(X_tr_tfidf.shape, y_train.shape)
print(X_cv_tfidf.shape, y_cv.shape)
print(X_te_tfidf.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix | tfidf
(49041, 18688) (49041,)
```

```
(24155, 18688) (24155,)
(36052, 18688) (36052,)
==========================================================================================================
```

After sugestion #5

In [123]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_tr_sug5_tfidf = hstack((X_train_text_tfidf, X_train_title_tfidf, X_train_state_ohe,
X_train_clean_ohe, X_train_cleanSub_ohe, X_train_grade_ohe, X_train_teacher_ohe,
X_train_prjResSum_ohe, X_train_quantity_norm, X_train_TprevPrj_norm,
X_train_price_norm,X_train_essay_wc_norm,X_train_essay_len_norm,X_train_title_wc_norm,X_train_title
_len_norm,X_train_prj_res_sum_wc_norm,X_train_prj_res_sum_len_norm)).tocsr()
X_cv_sug5_tfidf = hstack((X_cv_text_tfidf, X_cv_title_tfidf, X_cv_state_ohe, X_cv_clean_ohe, X_cv_c
leanSub_ohe, X_cv_grade_ohe, X_cv_teacher_ohe, X_cv_prjResSum_ohe, X_cv_quantity_norm, X_cv_TprevPr
j_norm,
X_cv_price_norm,X_cv_essay_wc_norm,X_cv_essay_len_norm,X_cv_title_wc_norm,X_cv_title_len_norm,X_cv_
prj_res_sum_wc_norm,X_cv_prj_res_sum_len_norm)).tocsr()
X_te_sug5_tfidf = hstack((X_test_text_tfidf, X_test_title_tfidf , X_test_state_ohe,
X_test_clean_ohe, X_test_cleanSub_ohe, X_test_grade_ohe, X_test_teacher_ohe, X_test_prjResSum_ohe,
X_test_quantity_norm, X_test_TprevPrj_norm,
X_test_price_norm,X_test_essay_wc_norm,X_test_essay_len_norm,X_test_title_wc_norm,X_test_title_len_
norm,X_test_prj_res_sum_wc_norm,X_test_prj_res_sum_len_norm)).tocsr()

print("Final Data matrix | TFIDF")
print(X_tr_sug5_tfidf.shape, y_train.shape)
print(X_cv_sug5_tfidf.shape, y_cv.shape)
print(X_te_sug5_tfidf.shape, y_test.shape)
print("="*100)
```

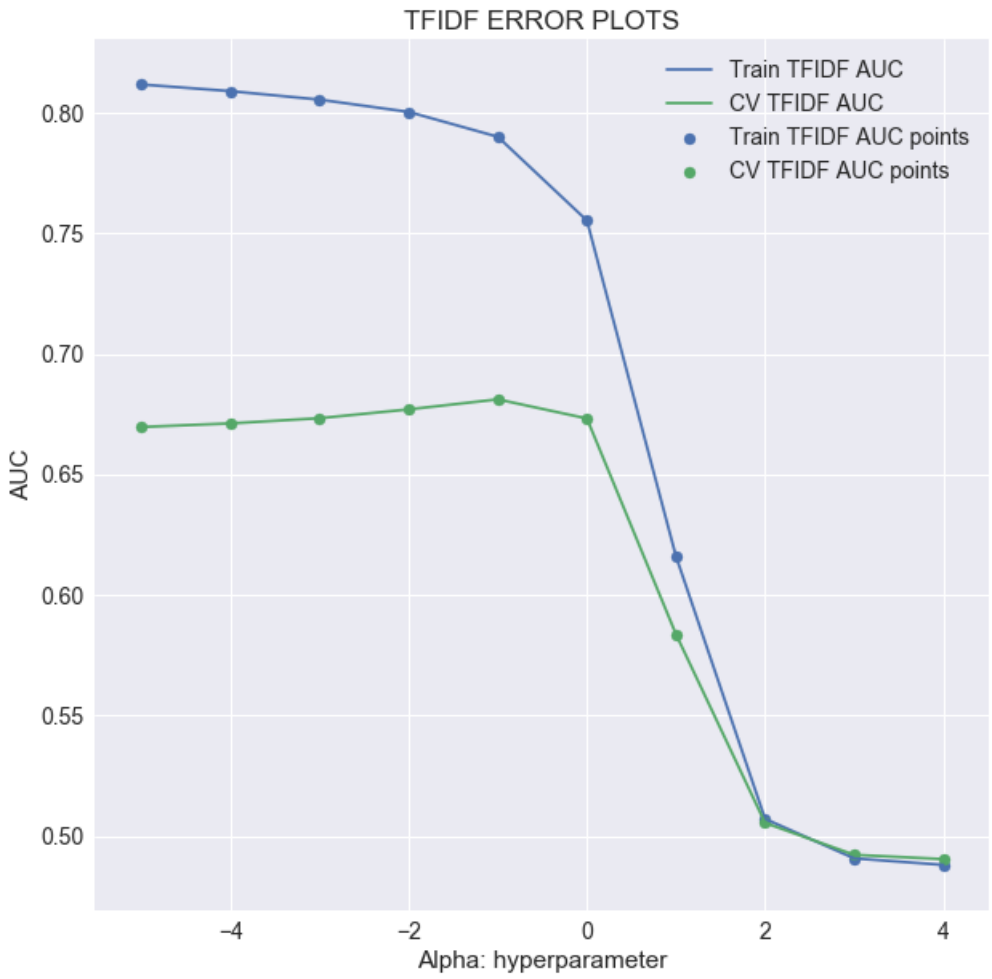
```
Final Data matrix | TFIDF
(49041, 18694) (49041,)
(24155, 18694) (24155,)
(36052, 18694) (36052,)
==========================================================================================================
```

In [124]:

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or no
n-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_tfidf_auc = []
cv_tfidf_auc = []

a = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0, 10, 100, 1000, 10000]
for i in a:
    clf = MultinomialNB(alpha = i)
    clf.fit(X_tr_tfidf, y_train)

    y_train_tfidf_pred = clf.predict_proba(X_tr_tfidf)[:,1]
    y_cv_tfidf_pred = clf.predict_proba(X_cv_tfidf)[:,1]

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html
```

```
    train_tfidf_auc.append(roc_auc_score(y_train,y_train_tfidf_pred))
    cv_tfidf_auc.append(roc_auc_score(y_cv, y_cv_tfidf_pred))


# change each element of list in its log value python
# https://stackoverflow.com/questions/47582264/python-how-to-convert-a-list-to-loglist
# #https://docs.scipy.org/doc/numpy/reference/generated/numpy.log10.html

from math import log
alpha = [np.log10(x) for x in a]
print(alpha)

#https://stackoverflow.com/questions/332289/how-do-you-change-the-size-of-figures-drawn-with-matplotlib
plt.figure(figsize=(10,10))
plt.plot(alpha, train_tfidf_auc, label='Train TFIDF AUC')
plt.plot(alpha, cv_tfidf_auc, label='CV TFIDF AUC')

plt.scatter(alpha, train_tfidf_auc, label='Train TFIDF AUC points')
plt.scatter(alpha, cv_tfidf_auc, label='CV TFIDF AUC points')
#plt.xscale('log')
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("TFIDF ERROR PLOTS")
plt.grid(True)
plt.show()
plt.close()
```

```
[-5.0, -4.0, -3.0, -2.0, -1.0, 0.0, 1.0, 2.0, 3.0, 4.0]
```

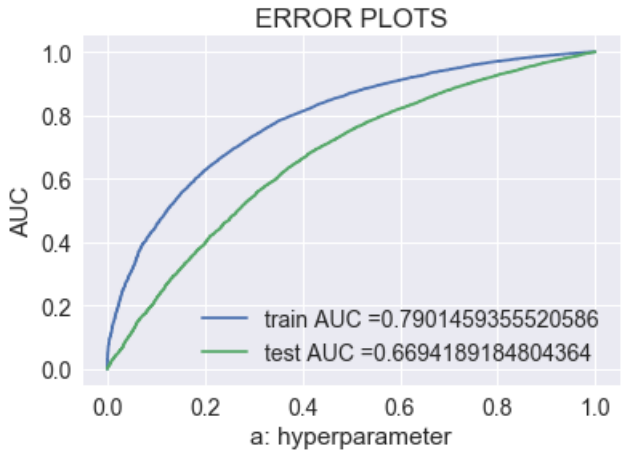


In [135]:

```
best_tfidf_a = 0.1
```

In [136]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = MultinomialNB(alpha = best_tfidf_a)
neigh.fit(X_tr_tfidf, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_tfidf_pred = neigh.predict_proba(X_tr_tfidf)[:,1]
y_test_tfidf_pred = neigh.predict_proba(X_te_tfidf)[:,1]

train_tfidf_fpr, train_tfidf_tpr, tr_tfidf_thresholds = roc_curve(y_train, y_train_tfidf_pred)
test_tfidf_fpr, test_tfidf_tpr, te_tfidf_thresholds = roc_curve(y_test, y_test_tfidf_pred)

plt.plot(train_tfidf_fpr, train_tfidf_tpr, label="train AUC ="+str(auc(train_tfidf_fpr, train_tfidf
_tpr)))
plt.plot(test_tfidf_fpr, test_tfidf_tpr, label="test AUC ="+str(auc(test_tfidf_fpr, test_tfidf_tpr)
))
plt.legend()
plt.xlabel("a: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```



In [137]:

```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Vectorizer", "Algorithm", "Hyper parameter", "Train AUC", "Test AUC"]

x.add_row(["TFIDF", "Naives Bayes", 0.01, 0.800461, 0.665206 ])
x.add_row(["TFIDF(Best)", "Naives Bayes", 0.1, 0.790145 , 0.669418 ])
x.add_row(["TFIDF", "Naives Bayes", 0, 0.820296, 0.648747 ])
x.add_row(["TFIDF", "Naives Bayes", 1, 0.7554001, 0.661907 ])
x.add_row(["TFIDF", "Naives Bayes", 10, 0.616066, 0.57516 ])

print(x)
```

```
+-------------+-------------+-----------------+-----------+----------+
|  Vectorizer |  Algorithm  | Hyper parameter | Train AUC | Test AUC |
+-------------+-------------+-----------------+-----------+----------+
|    TFIDF    | Naives Bayes |       0.01      |  0.800461 | 0.665206 |
| TFIDF(Best) | Naives Bayes |       0.1       |  0.790145 | 0.669418 |
|    TFIDF    | Naives Bayes |        0        |  0.820296 | 0.648747 |
|    TFIDF    | Naives Bayes |        1        | 0.7554001 | 0.661907 |
|    TFIDF    | Naives Bayes |        10       |  0.616066 | 0.57516  |
+-------------+-------------+-----------------+-----------+----------+
```

In [138]:

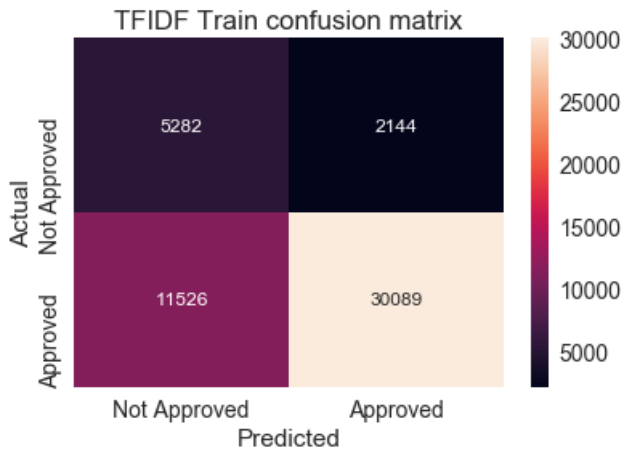
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_tfidf_pred, tr_tfidf_thresholds, train_tfidf_fpr, t
rain_tfidf_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_tfidf_pred, te_tfidf_thresholds, test_tfidf_fpr, test
_tfidf_tpr)))
```

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5158032415891908 for threshold 0.854
[[ 5282  2144]
 [11526 30089]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3994874364299064 for threshold 0.922
[[ 3587  1872]
 [12382 18211]]
```

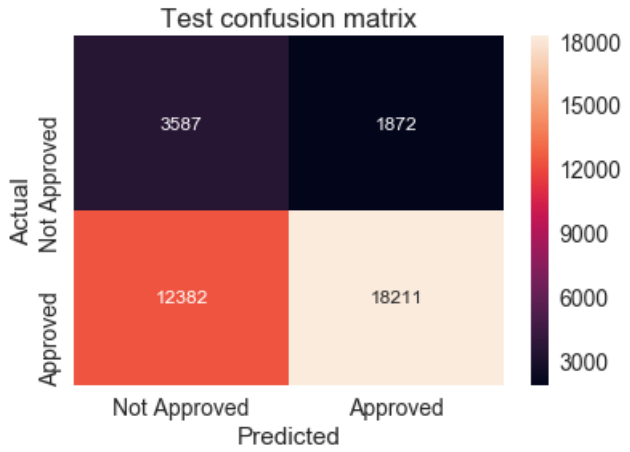
In [139]:

```
import seaborn as snTr
import seaborn as snTe
import pandas as pdH
import matplotlib.pyplot as pltTr
import matplotlib.pyplot as pltTe

# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
array=confusion_matrix(y_train, predict(y_train_tfidf_pred, tr_tfidf_thresholds, train_tfidf_fpr, t
rain_tfidf_tpr))
df_cmTr = pdH.DataFrame(array,range(2),range(2))

# https://stackoverflow.com/questions/32723798/how-do-i-add-a-title-to-seaborn-heatmap
axTr = pltTr.axes()
snTr.set(font_scale=1.4)#for label size

# https://seaborn.pydata.org/generated/seaborn.heatmap.html
snTr.heatmap(df_cmTr, annot=True,annot_kws={"size": 12},fmt="d",ax=axTr)# font size, format in
digit

labels=['Not Approved','Approved']
axTr.set_xticklabels(labels)
axTr.set_yticklabels(labels)
pltTr.title("TFIDF Train confusion matrix")
pltTr.xlabel("Predicted")
pltTr.ylabel("Actual")
pltTr.show()

# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
arrayTe=confusion_matrix(y_test, predict(y_test_tfidf_pred, te_tfidf_thresholds, test_tfidf_fpr, te
st_tfidf_tpr))
df_cmTe = pdH.DataFrame(arrayTe,range(2),range(2))

# https://stackoverflow.com/questions/50947776/plot-two-seaborn-heatmap-graphs-side-by-side
#fig, ax =plt.subplots(1,1)

axTe = plt.axes()
snTe.set(font_scale=1.4)#for label size

# https://seaborn.pydata.org/generated/seaborn.heatmap.html
snTe.heatmap(df_cmTe, annot=True,annot_kws={"size": 12},fmt="d",ax=axTe)# font size, format in
digit

labels=['Not Approved','Approved']
axTe.set_xticklabels(labels)
axTe.set_yticklabels(labels)
pltTe.title("Test confusion matrix")
pltTe.xlabel("Predicted")
pltTe.ylabel("Actual")
pltTe.show()
```

```
the maximum value of tpr*(1-fpr) 0.5158032415891908 for threshold 0.854
```




```
the maximum value of tpr*(1-fpr) 0.3994874364299064 for threshold 0.922
```



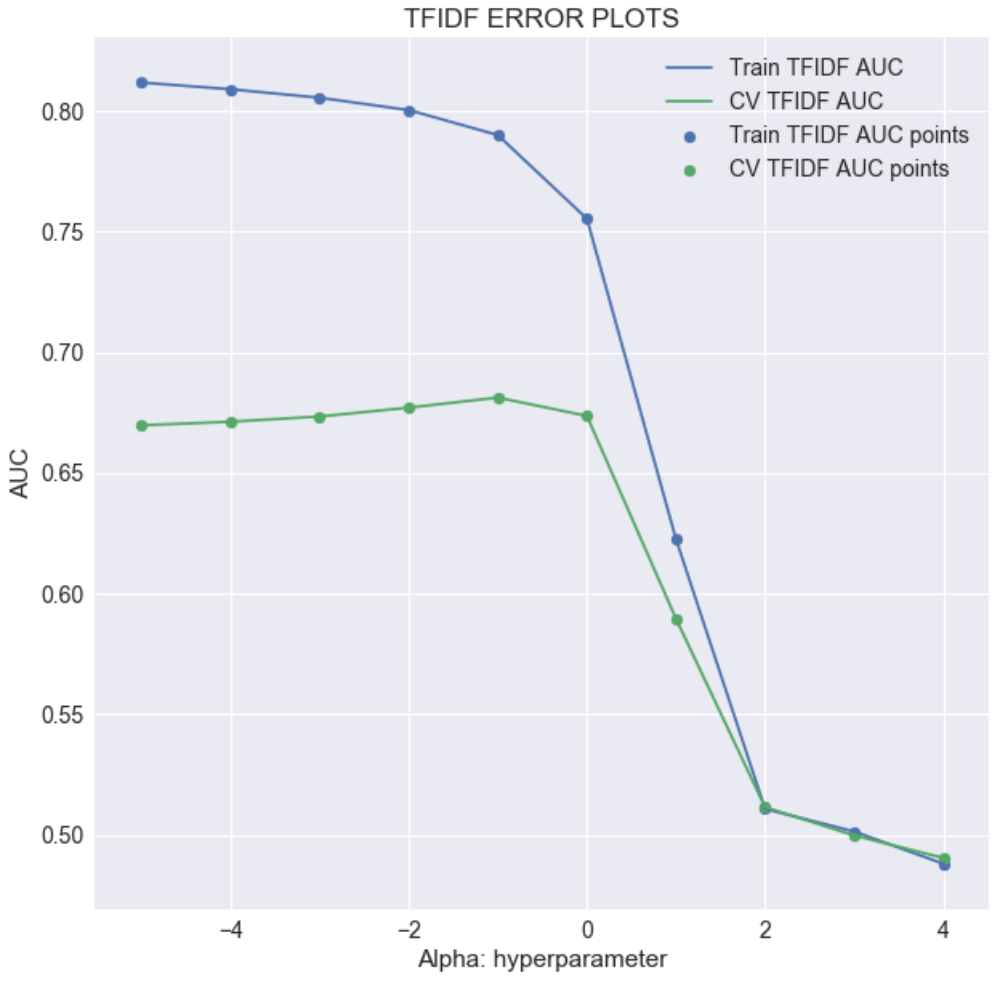
Suggestion #5

In [140]:

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or no
n-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_sug5_tfidf_auc = []
cv_sug5_tfidf_auc = []

a = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0, 10, 100, 1000, 10000]
for i in a:
    clf = MultinomialNB(alpha = i)
    clf.fit(X_tr_sug5_tfidf, y_train)

    y_train_sug5_tfidf_pred = clf.predict_proba(X_tr_sug5_tfidf)[:,1]
    y_cv_sug5_tfidf_pred = clf.predict_proba(X_cv_sug5_tfidf)[:,1]

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
```

```
    # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html
    train_sug5_tfidf_auc.append(roc_auc_score(y_train,y_train_sug5_tfidf_pred))
    cv_sug5_tfidf_auc.append(roc_auc_score(y_cv, y_cv_sug5_tfidf_pred))


# change each element of list in its log value python
# https://stackoverflow.com/questions/47582264/python-how-to-convert-a-list-to-loglist
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.log10.html

from math import log
alpha = [np.log10(x) for x in a]
print(alpha)

#https://stackoverflow.com/questions/332289/how-do-you-change-the-size-of-figures-drawn-with-matplotlib
plt.figure(figsize=(10,10))
plt.plot(alpha, train_sug5_tfidf_auc, label='Train TFIDF AUC')
plt.plot(alpha, cv_sug5_tfidf_auc, label='CV TFIDF AUC')

plt.scatter(alpha, train_sug5_tfidf_auc, label='Train TFIDF AUC points')
plt.scatter(alpha, cv_sug5_tfidf_auc, label='CV TFIDF AUC points')
#plt.xscale('log')
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("TFIDF ERROR PLOTS")
plt.grid(True)
plt.show()
plt.close()
```

```
[-5.0, -4.0, -3.0, -2.0, -1.0, 0.0, 1.0, 2.0, 3.0, 4.0]
```

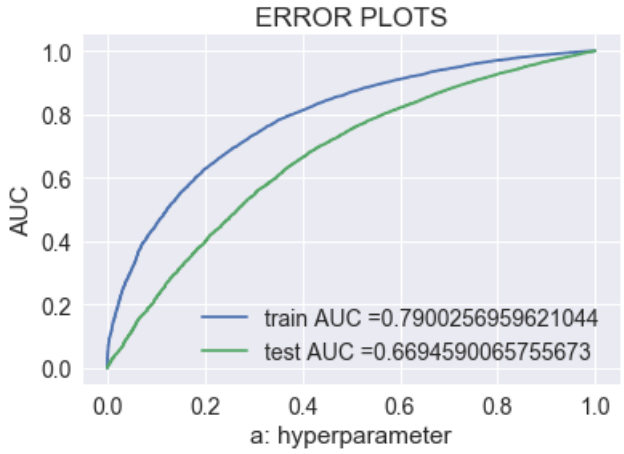


In [151]:

```
best_tfidf_a = 0.1
```

In [152]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = MultinomialNB(alpha = best_tfidf_a)
neigh.fit(X_tr_sug5_tfidf, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_sug5_tfidf_pred = neigh.predict_proba(X_tr_sug5_tfidf)[:,1]
y_test_sug5_tfidf_pred = neigh.predict_proba(X_te_sug5_tfidf)[:,1]

train_tfidf_fpr, train_tfidf_tpr, tr_tfidf_thresholds = roc_curve(y_train, y_train_sug5_tfidf_pred
)
test_tfidf_fpr, test_tfidf_tpr, te_tfidf_thresholds = roc_curve(y_test, y_test_sug5_tfidf_pred)

plt.plot(train_tfidf_fpr, train_tfidf_tpr, label="train AUC ="+str(auc(train_tfidf_fpr, train_tfidf
_tpr)))
plt.plot(test_tfidf_fpr, test_tfidf_tpr, label="test AUC ="+str(auc(test_tfidf_fpr, test_tfidf_tpr)
))
plt.legend()
plt.xlabel("a: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```



In [153]:

```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Vectorizer", "Algorithm", "Hyper parameter", "Train AUC", "Test AUC"]

x.add_row(["TFIDF (added feature)", "Naives Bayes", 0.01, 0.800373 , 0.6652274 ])
x.add_row(["TFIDF (added feature)(Best)", "Naives Bayes", 0.1, 0.790025 , 0.66945 ])
x.add_row(["TFIDF (added feature)", "Naives Bayes", 0, 0.82022 , 0.64875 ])
x.add_row(["TFIDF (added feature)", "Naives Bayes", 1, 0.755354, 0.66227 ])
x.add_row(["TFIDF (added feature)", "Naives Bayes", 10, 0.622575 , 0.58067 ])

print(x)
```

```
+------------------------------+--------------+-----------------+-----------+-----------+
|          Vectorizer          |  Algorithm   | Hyper parameter | Train AUC |  Test AUC |
+------------------------------+--------------+-----------------+-----------+-----------+
|    TFIDF (added feature)     | Naives Bayes |       0.01      |  0.800373 | 0.6652274 |
| TFIDF (added feature)(Best)  | Naives Bayes |       0.1       |  0.790025 |  0.66945  |
|    TFIDF (added feature)     | Naives Bayes |        0        |  0.82022  |  0.64875  |
|    TFIDF (added feature)     | Naives Bayes |        1        |  0.755354 |  0.66227  |
|    TFIDF (added feature)     | Naives Bayes |        10       |  0.622575 |  0.58067  |
+------------------------------+--------------+-----------------+-----------+-----------+
```

In [154]:

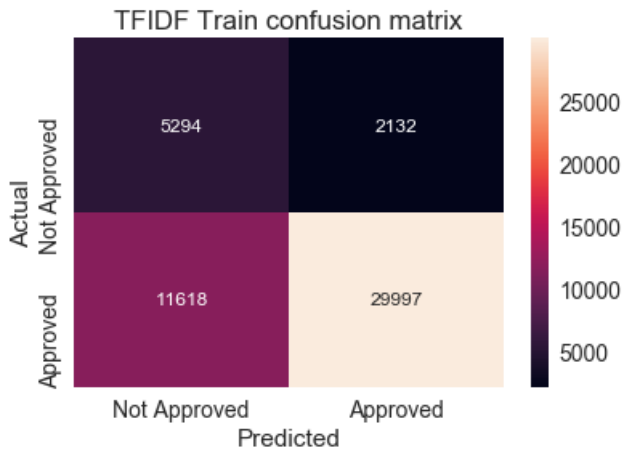
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_sug5_tfidf_pred, tr_tfidf_thresholds,
train_tfidf_fpr, train_tfidf_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_sug5_tfidf_pred, te_tfidf_thresholds, test_tfidf_fpr,
test_tfidf_tpr)))
```

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5157846222178416 for threshold 0.856
[[ 5294  2132]
 [11618 29997]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3993085998149289 for threshold 0.92
[[ 3573  1886]
 [12279 18314]]
```

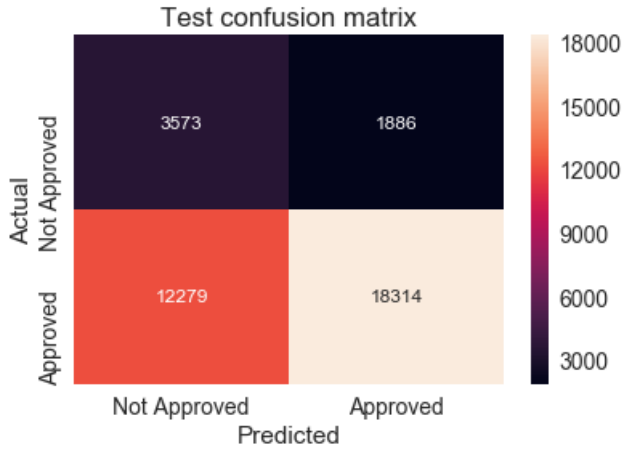
In [155]:

```
import seaborn as snTr
import seaborn as snTe
import pandas as pdH
import matplotlib.pyplot as pltTr
import matplotlib.pyplot as pltTe

# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
array=confusion_matrix(y_train, predict(y_train_sug5_tfidf_pred, tr_tfidf_thresholds,
train_tfidf_fpr, train_tfidf_tpr))
df_cmTr = pdH.DataFrame(array,range(2),range(2))

# https://stackoverflow.com/questions/32723798/how-do-i-add-a-title-to-seaborn-heatmap
axTr = pltTr.axes()
snTr.set(font_scale=1.4)#for label size

# https://seaborn.pydata.org/generated/seaborn.heatmap.html
snTr.heatmap(df_cmTr, annot=True,annot_kws={"size": 12},fmt="d",ax=axTr)# font size, format in
digit

labels=['Not Approved','Approved']
axTr.set_xticklabels(labels)
axTr.set_yticklabels(labels)
pltTr.title("TFIDF Train confusion matrix")
pltTr.xlabel("Predicted")
pltTr.ylabel("Actual")
pltTr.show()

# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
arrayTe=confusion_matrix(y_test, predict(y_test_sug5_tfidf_pred, te_tfidf_thresholds,
test_tfidf_fpr, test_tfidf_tpr))
df_cmTe = pdH.DataFrame(arrayTe,range(2),range(2))

# https://stackoverflow.com/questions/50947776/plot-two-seaborn-heatmap-graphs-side-by-side
#fig, ax =plt.subplots(1,1)

axTe = plt.axes()
snTe.set(font_scale=1.4)#for label size

# https://seaborn.pydata.org/generated/seaborn.heatmap.html
snTe.heatmap(df_cmTe, annot=True,annot_kws={"size": 12},fmt="d",ax=axTe)# font size, format in
digit

labels=['Not Approved','Approved']
axTe.set_xticklabels(labels)
axTe.set_yticklabels(labels)
pltTe.title("Test confusion matrix")
pltTe.xlabel("Predicted")
pltTe.ylabel("Actual")
pltTe.show()
```

```
the maximum value of tpr*(1-fpr) 0.5157846222178416 for threshold 0.856
```




```
the maximum value of tpr*(1-fpr) 0.3993085998149289 for threshold 0.92
```



2.4.2.1 Top 10 important features of positive class from SET 2

In [156]:

```
# Please write all the code with proper documentation
```

In [157]:

```
print(type(ii))
print(type(j))
```

```
<class 'list'>
<class 'list'>
```

In [158]:

```
#print(len(aa))
#print(len(b))
#print(len(c))
#print(len(d))
#print(len(e))
#print(len(f))
#print(len(h))
#print(len(ii))
#print(len(j))
#print(len(l))
```

In [159]:

```
feature_list_set2=[]
print(feature_list_set2)
#print(i)
print(type(i))
feature_list_set2=aa+b+c+d+e+f+h+ii+j+l
#print(feature_list_set2)
print(len(feature_list_set2))
```

```
[]
<class 'int'>
18694
```

In [160]:

```
pos_class_TFIDF_prob_sorted = neigh.feature_log_prob_[1, :].argsort()

# argsort() will return the indices of values from low probability to high probability.
# When you print feature names of these indices, these indices will return you the feature names w
ith low probability.
# So, please reverse the indices after argsort()
```

In [161]:

```
print(len(feature_list_set2))
print(neigh.feature_log_prob_[0, :].shape)
print(len(pos_class_TFIDF_prob_sorted))
```

```
18694
(18694,)
18694
```

In [162]:

```
print(pos_class_TFIDF_prob_sorted)  # in low probablity to high
print(pos_class_TFIDF_prob_sorted[::-1]) # reverse ; now from high probablity to low

pos_class_TFIDF_prob_sorted=pos_class_TFIDF_prob_sorted[::-1]
print(pos_class_TFIDF_prob_sorted)
```

```
[14691 14688 14690 ... 17060 18328 18134]
[18134 18328 17060 ... 14690 14688 14691]
[18134 18328 17060 ... 14690 14688 14691]
```

In [163]:

```
# https://imgur.com/a/1QObgQ2
# https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-bayes
print(np.take(feature_list_set2, pos_class_TFIDF_prob_sorted[:10]))
```

```
['refugees' 'spin' 'de' 'destination' 'prj_res_sum_len' 'prj_res_sum_wc'
 'title_len' 'title_wc' 'essay_len' 'essay_wc']
```

Suggestion

In you data, you will be knowing hstacked features and their dimensions. Check the word in dimension and print that word. Suppose you have text+essay+categorial let text have dimension 120, essay 120, categorical 10 hstack in this way (text+essay+categorical) 120+120+10=250 dimensions Let important word indices be 10,196,243. for index 10 search in text , 196 in essay, 243 in categorical.

In [164]:

```
# https://cmdlinetips.com/2018/01/how-to-create-pandas-dataframe-from-multiple-lists/
Pos_TFIDF_Feature_dataFrame=pd.DataFrame({'Feature Word': feature_list_set2,'Feature_Probablity' :
neigh.feature_log_prob_[1, :]})
```

In [165]:

```
Pos_TFIDF_Feature_dataFrame.head(10)
```

Out[165]:

	Feature Word	Feature_Probablity
0	ak	-12.006352
1	al	-11.003842
2	ar	-9.851227
3	az	-9.373898
4	ca	-12.305504
5	co	-13.578395
6	ct	-13.363665
7	dc	-13.716568
8	de	-13.839003
9	fl	-14.091449

In [166]:

```
# https://cmdlinetips.com/2018/02/how-to-sort-pandas-dataframe-by-columns-and-row/
Pos_TFIDF_Feature_dataFrame_Sorted=Pos_TFIDF_Feature_dataFrame.sort_values('Feature_Probablity',as
cending=False)
Pos_TFIDF_Feature_dataFrame_Sorted.head(11)
```

Out[166]:

	Feature Word	Feature_Probablity
18134	refugees	-3.715055
18328	spin	-3.728605
17060	de	-3.748131
17079	destination	-3.752124
18693	prj_res_sum_len	-3.802040
18685	yourself	-3.802040
18692	prj_res_sum_wc	-3.802040
18691	title_len	-3.802040
18690	title_wc	-3.802040
18689	essay_len	-3.802040
18688	essay_wc	-3.802040

2.4.2.2 Top 10 important features of negative class from SET 2

In [167]:

```
# Please write all the code with proper documentation
```

In [168]:

```
neg_class_TFIDF_prob_sorted = neigh.feature_log_prob_[0, :].argsort()

# argsort() will return the indices of values from low probability to high probability.
# When you print feature names of these indices, these indices will return you the feature names w
ith low probability.
# So, please reverse the indices after argsort()
```

In [169]:

```
print(neg_class_TFIDF_prob_sorted)  # in low probablity to high
print(neg_class_TFIDF_prob_sorted[::-1]) # reverse ; now from high probablity to low

neg_class_TFIDF_prob_sorted=neg_class_TFIDF_prob_sorted[::-1]
print(neg_class_TFIDF_prob_sorted)
```

```
[10315  4472  6340 ... 17079 18134 18328]
[18328 18134 17079 ...  6340  4472 10315]
[18328 18134 17079 ...  6340  4472 10315]
```

In [170]:

```
# https://imgur.com/a/1QObgQ2
# https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-bayes
print(np.take(feature_list_set2, neg_class_TFIDF_prob_sorted[:10]))
```

```
['spin' 'refugees' 'destination' 'de' 'yourself' 'prj_res_sum_wc'
 'prj_res_sum_len' 'zone' 'essay_wc' 'essay_len']
```

Suggestion

In you data, you will be knowing hstacked features and their dimensions. Check the word in dimension and print that word. Suppose you have text+essay+categorial let text have dimension 120, essay 120, categorical 10 hstack in this way (text+essay+categorical) 120+120+10=250 dimensions Let important word indices be 10,196,243. for index 10 search in text , 196 in essay, 243 in categorical.

In [171]:

```
# https://cmdlinetips.com/2018/01/how-to-create-pandas-dataframe-from-multiple-lists/
Neg_TFIDF_Feature_dataFrame=pd.DataFrame({'Feature Word': feature_list_set2,'Feature_Probablity' :
neigh.feature_log_prob_[0, :]})
```

In [172]:

```
print(Neg_TFIDF_Feature_dataFrame.head(5))
print(Neg_TFIDF_Feature_dataFrame.tail(5))
```

```
  Feature Word  Feature_Probablity
0           ak          -12.021021
1           al          -11.002851
2           ar          -10.077034
3           az           -9.301758
4           ca          -11.853734
          Feature Word  Feature_Probablity
18689        essay_len           -3.817496
18690         title_wc           -3.817496
18691        title_len           -3.817496
18692   prj_res_sum_wc           -3.817496
18693  prj_res_sum_len           -3.817496
```

In [173]:

```
# https://cmdlinetips.com/2018/02/how-to-sort-pandas-dataframe-by-columns-and-row/
Neg_TFIDF_Feature_dataFrame_Sorted=Neg_TFIDF_Feature_dataFrame.sort_values('Feature_Probablity',as
cending=False)
Neg_TFIDF_Feature_dataFrame_Sorted.head(11)
```

Out[173]:

	Feature Word	Feature_Probablity
18328	spin	-3.692469
18134	refugees	-3.713849
17079	destination	-3.754230
17060	de	-3.754483

	Feature Word	Feature_Probablity
18689	essay_len	-3.817496
18685	yourself	-3.817496
18687	zone	-3.817496
18688	essay_wc	-3.817496
18693	prj_res_sum_len	-3.817496
18690	title_wc	-3.817496
18691	title_len	-3.817496

3. Conclusions

In [174]:

```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Vectorizer", "Algorithm", "Hyper parameter", "AUC"]

x.add_row(["BOW", "Naives Bayes", 1, 0.71314 ])
x.add_row(["BOW(added feature)", "Naives Bayes", 1, 0.710635])
x.add_row(["TFIDF","Naives Bayes",0.1 , 0.669418 ])
x.add_row(["TFIDF(added feature)","Naives Bayes", 0.1, 0.66945 ])

print(x)
```

```
+----------------------+--------------+-----------------+----------+
|      Vectorizer      |  Algorithm   | Hyper parameter |   AUC    |
+----------------------+--------------+-----------------+----------+
|         BOW          | Naives Bayes |        1        | 0.71314  |
|  BOW(added feature)  | Naives Bayes |        1        | 0.710635 |
|        TFIDF         | Naives Bayes |       0.1       | 0.669418 |
| TFIDF(added feature) | Naives Bayes |       0.1       | 0.66945  |
+----------------------+--------------+-----------------+----------+
```

Step followed

- Preprocessing of Project_subject_categories and Project_subject_subcategories
- Preprocessing of Project_essay and Project_title

```
- (SUGGESTION#5) took column for no of words in essay
- (SUGGESTION#5) took column for lenght of each cell in essay
- (SUGGESTION#5) took column for no of words in Title
- (SUGGESTION#5) took column for lenght of each cell in Title
- (SUGGESTION#5) took column for no of words in Project resource summary
- (SUGGESTION#5) took column for lenght of each cell in Project resource su
mmary
```

- Vectorization(one hot encoding) for clean_category, clean_subcategory, teacher_prefix
- Vectorization(one hot encoding) for BOW(project_essay and project_title) for TFIDF(project_essay and project _title)
- Vectorizing Numeric features (Standardization of Price column)
- (SUGGESTION#5) Added all the six features (word count and lenght of essay,title and project_resource_summary) to project_data
- ~~Took first 50000 data points for doing the assignment and removed the Class lable (Project_is_approved)~~
- Took data points for doing the assignment and separate the Class lable (Project_is_approved)
- Splitting Data into Train (further split into Train and Cross validation) and Test.
- Making datamodel ready

text

```
- encoding of school_state is splited into Train,CV and Test vector
                and stored the feature name in aa
```

```
                - encoding of clean_category is splited into Train,CV and Test vector
                                and stored the feature name in b
                - encoding of clean_subcategory is splited into Train,CV and Test vector
                                and stored the feature name in c
                - encoding of project_grade_category is splited into Train,CV and Test vect
    or
                                and stored the feature name in d
                - encoding of teacher_prefix is splited into Train,CV and Test vector
                                and stored the feature name in e
                - encoding of project_resource_summary is splited into Train,CV and Test ve
    ctor
                                and stored the feature name in f
                - encoding of project_essay(BOW) is splited into Train,CV and Test vector
                                and stored the feature name in g
                - encoding of project_title(BOW) is splited into Train,CV and Test vector
                                and stored the feature name in k
```

numeric

```
            - encoding of price is splited into Train,CV and Test vector
            - encoding of teacher_number_of_previously_posted_projects is splited into Train
,CV and Test vector
            - encoding of quantity is splited into Train,CV and Test vector
```

SUGGESTION#5 - encoding of essay_words is splited into Train,CV and Test vector

```
            - encoding of essay_length is splited into Train,CV and Test vector
            - encoding of title_words is splited into Train,CV and Test vector
            - encoding of title_length is splited into Train,CV and Test vector
            - encoding of prj_summ_resource_words is splited into Train,CV and Test vector
            - encoding of prj_summ_resource_length is splited into Train,CV and Test vector
```

- Adding all the features
  - Numeric features are stored in ('price','quantity','teacher_number_of_previously_posted_projects') as h
  - Numeric features are stored in essay_wc','essay_len','title_wc','title_len','prj_res_sum_wc','prj_res_sum_len') as l
  - in Feature_list, added all the text lists and numeric list (a,b,c,d,e,f,g,k,l and h)

For SET 1

Merging all the above features for SET 1

- Horizontally merging(with hstack) all categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)
- Horizontally merging(with hstack) all categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW) and six SUGGESTION#5 features

Hyper paramter tuning to find best alpha

SUGGESTION#1 ~~define batch_predict() function, which will takes Classifier and data as input~~

- SUGGESTION#2 ~~1 Draw a graph for different value of alpha, between Train and CV of BOW data (SET 1)~~
- Draw a graph for different value of alpha, using log(alpha), between Train and CV of BOW data (SET 1)
- Choose the Best alpha, by seeing which alpha gives better TEST AUC
- Made pretty table, showing various value of Alpha
- Draw AUC for Train and Test data
- Create Confusion matrix, in heatmap.
- SUGGESTION#4, added xlabel and ylables in confusion matrix

- SUGGESTION#5 Draw a graph for different value of alpha, between Train and CV of BOW data (SET 1), having six new numeric features.

- Choose the Best alpha, by seeing which alpha gives better TEST AUC
- Made pretty table, showing various value of Alpha

- Draw AUC for Train and Test data
- Create Confusion matrix, in heatmap.
- SUGGESTION#4, added xlabel and ylables in confusion matrix

Top 10 important features of positive class from SET 1

take the top 10 positive feature, my mapping, probablity (neigh.feature_log*prob*[0, :] in sorted) and feature name SUGGESTION#3, reverse the indices, since argsort() will return the indices of values from low probability to high probability.

Top 10 important features of negative class from SET 1

take the top 10 negative feature, my mapping, probablity (neigh.feature_log*prob*[1, :] in sorted) and feature name SUGGESTION#3, reverse the indices, since argsort() will return the indices of values from low probability to high probability.

For SET 2

```
- encoding of project_essay(TFIDF) is splited into Train,CV and Test vector
                and stored the feature name in i
- encoding of project_title(TFIDF) is splited into Train,CV and Test vector
                and stored the feature name in j
```

- Merging all the text and numeric features for SET 2

- Horizontally merging(with hstack) all categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF)
- SUGGESTION#5 Horizontally merging(with hstack) all categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF) and six suggestion#5 features

- Hyper paramter tuning to find best alpha

```
1 Draw a graph for different value of alpha, between Train and CV of TFIDF data
(SET 2)
2 Choose the Best alpha, by seeing which alpha gives better TEST AUC
3 Draw AUC for Train and Test data
4 Create Confusion matrix, in heatmap.
```

- Hyper paramter tuning to find best alpha

1 - SUGGESTION#2 ~~1 Draw a graph for different value of alpha, between Train and CV of TFIDF data (SET 2)~~

```
1 Draw a graph for different value of alpha, using log(alpha), between Train and
CV of TFIDF data (SET 2)
2 Choose the Best alpha, by seeing which alpha gives better TEST AUC
<font color="green"> Made pretty table, showing various value of Alpha </font>
3 Draw AUC for Train and Test data
4 Create Confusion matrix, in heatmap.
SUGGESTION#4, added xlabel and ylables in confusion matrix

SUGGESTION#5
1 Draw a graph for different value of alpha, between Train and CV of BOW data (S
ET 1), having six new numeric features.
2 Choose the Best alpha, by seeing which alpha gives better TEST AUC
```

Made pretty table, showing various value of Alpha

```
3 Draw AUC for Train and Test data
4 Create Confusion matrix, in heatmap.
SUGGESTION#4, added xlabel and ylables in confusion matrix
```

- Adding all the features for TFIDF

```
- in Feature_list_set2, added all the text lists and numeric list
(a,b,c,d,e,f,i,j,l and h)
```

Top 10 important features of positive class from SET 2

take the top 10 positive feature, my mapping, probablity (neigh.feature_log*prob*[1, :] in sorted) and feature name SUGGESTION#3, reverse the indices, since argsort() will return the indices of values from low probability to high probability.

Top 10 important features of negative class from SET 2

take the top 10 negative feature, my mapping, probablity (neigh.feature_log*prob*[0, :] in sorted) and feature name SUGGESTION#3, reverse the indices, since argsort() will return the indic